SAP HANA SQL Reference Guide for SAP HANA Platform
# Content

1. SAP HANA SQL Reference Guide for SAP HANA Platform. ........................................... 21
2. SAP HANA SQL Reference Guide (New and Changed). ........................................... 22
3. Introduction. .................................................................................................................. 26
4. SQL Reference. .............................................................................................................. 27
   4.1 Introduction to SQL. ................................................................................................. 27
   4.2 SQL Notation Conventions. ..................................................................................... 30
   4.3 Data Types. .............................................................................................................. 33
      - Binary Data Types. ............................................................................................... 35
      - Boolean Data Type. .............................................................................................. 35
      - Character String Data Types. .............................................................................. 36
      - Data Type Conversion. ........................................................................................................ 37
      - Datetime Data Types. ............................................................................................ 40
      - Large Object (LOB) Data Types. ........................................................................ 45
      - Multi-valued Data Types. ....................................................................................... 47
      - Numeric Data Types. .............................................................................................. 48
      - Spatial Data Types. .................................................................................................. 49
   4.4 Reserved Words. .......................................................................................................... 50
   4.5 Operators. .................................................................................................................... 51
   4.6 Expressions. .............................................................................................................. 56
   4.7 Predicates. ................................................................................................................... 61
      - Comparison Predicates (ANY, SOME, and ALL). .................................................. 62
      - BETWEEN Predicate. .............................................................................................. 63
      - CONTAINS Predicate. ............................................................................................ 64
      - EXISTS Predicate. .................................................................................................... 68
      - IN Predicate. ............................................................................................................ 69
      - LIKE Predicate. ....................................................................................................... 71
      - LIKE_REGEXPR Predicate. .................................................................................... 72
      - MEMBER OF Predicate. ......................................................................................... 73
      - NULL Predicate. ....................................................................................................... 74
   4.8 Session Variables. ........................................................................................................ 75
   4.9 SQL Functions. ............................................................................................................ 83
      - Alphabetical List Of Functions. .............................................................................. 84
      - Aggregate Functions. ............................................................................................. 414
      - Array Functions. ...................................................................................................... 415
      - Data Type Conversion Functions. ........................................................................... 416
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datetime Functions</td>
<td>417</td>
</tr>
<tr>
<td>Fulltext Functions</td>
<td>418</td>
</tr>
<tr>
<td>Hierarchy Functions</td>
<td>418</td>
</tr>
<tr>
<td>JSON Functions</td>
<td>419</td>
</tr>
<tr>
<td>Miscellaneous Functions</td>
<td>420</td>
</tr>
<tr>
<td>Numeric Functions</td>
<td>421</td>
</tr>
<tr>
<td>Security Functions</td>
<td>422</td>
</tr>
<tr>
<td>Series Data Functions</td>
<td>422</td>
</tr>
<tr>
<td>String Functions</td>
<td>423</td>
</tr>
<tr>
<td>Window Functions and the Window Specification</td>
<td>424</td>
</tr>
<tr>
<td>SQL Statements</td>
<td>430</td>
</tr>
<tr>
<td>Alphabetical List of Statements</td>
<td>431</td>
</tr>
<tr>
<td>Access Control Statements</td>
<td>1199</td>
</tr>
<tr>
<td>Backup and Recovery Statements</td>
<td>1200</td>
</tr>
<tr>
<td>Data Definition Statements</td>
<td>1201</td>
</tr>
<tr>
<td>Data Import Export Statements</td>
<td>1202</td>
</tr>
<tr>
<td>Data Manipulation Statements</td>
<td>1203</td>
</tr>
<tr>
<td>Client-side Encryption Statements</td>
<td>1203</td>
</tr>
<tr>
<td>JSON Document Store Statements</td>
<td>1204</td>
</tr>
<tr>
<td>Procedural Statements</td>
<td>1204</td>
</tr>
<tr>
<td>Session Management Statements</td>
<td>1205</td>
</tr>
<tr>
<td>System Management Statements</td>
<td>1205</td>
</tr>
<tr>
<td>Tenant Database Management Statements</td>
<td>1207</td>
</tr>
<tr>
<td>Transaction Management Statements</td>
<td>1208</td>
</tr>
<tr>
<td>Workload Management Statements</td>
<td>1208</td>
</tr>
<tr>
<td>System Limitations</td>
<td>1208</td>
</tr>
<tr>
<td>SQL Error Codes</td>
<td>1212</td>
</tr>
<tr>
<td>ANSI SQL Compliance</td>
<td>1272</td>
</tr>
<tr>
<td>Compliance With Mandatory ANSI SQL Features</td>
<td>1272</td>
</tr>
<tr>
<td>SQL Reference for Additional SAP HANA Contexts</td>
<td>1283</td>
</tr>
<tr>
<td>Accelerator for SAP ASE</td>
<td>1283</td>
</tr>
<tr>
<td>ALTER DATABASE Statement [Accelerator for SAP ASE]</td>
<td>1284</td>
</tr>
<tr>
<td>ALTER SYSTEM ALTER CONFIGURATION Statement [Accelerator for SAP ASE]</td>
<td>1285</td>
</tr>
<tr>
<td>ALTER SYSTEM INITIALIZE SERVICE Statement [Accelerator for SAP ASE]</td>
<td>1287</td>
</tr>
<tr>
<td>ALTER SYSTEM UNINITIALIZE SERVICE Statement [Accelerator for SAP ASE]</td>
<td>1288</td>
</tr>
<tr>
<td>Dynamic Tiering</td>
<td>1289</td>
</tr>
<tr>
<td>ALTER AUDIT POLICY Statement [Dynamic Tiering]</td>
<td>1292</td>
</tr>
<tr>
<td>ALTER DATABASE Statement [Dynamic Tiering]</td>
<td>1294</td>
</tr>
<tr>
<td>ALTER EXTENDED STORAGE Statement [Dynamic Tiering]</td>
<td>1295</td>
</tr>
<tr>
<td>ALTER INDEX Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1298</td>
</tr>
<tr>
<td>ALTER STATISTICS Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1299</td>
</tr>
<tr>
<td>Statement</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>ALTER STATISTICS Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1305</td>
</tr>
<tr>
<td>ALTER SYSTEM ALTER CONFIGURATION Statement [Dynamic Tiering]</td>
<td>1312</td>
</tr>
<tr>
<td>ALTER SYSTEM RECONFIGURE SERVICE Statement [Dynamic Tiering]</td>
<td>1318</td>
</tr>
<tr>
<td>ALTER TABLE Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1320</td>
</tr>
<tr>
<td>ALTER TABLE Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1326</td>
</tr>
<tr>
<td>CALL CHECK_CATALOG Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1340</td>
</tr>
<tr>
<td>CALL CHECK_ES Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1343</td>
</tr>
<tr>
<td>CALL CHECK_TABLE_CONSISTENCY Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1346</td>
</tr>
<tr>
<td>CALL ES_REBUILD_INDEX Statement (Multistore Tables) [Dynamic Tiering]</td>
<td>1349</td>
</tr>
<tr>
<td>CREATE AUDIT POLICY Statement [Dynamic Tiering]</td>
<td>1350</td>
</tr>
<tr>
<td>CREATE EXTENDED STORAGE Statement [Dynamic Tiering]</td>
<td>1355</td>
</tr>
<tr>
<td>CREATE INDEX Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1357</td>
</tr>
<tr>
<td>CREATE INDEX Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1359</td>
</tr>
<tr>
<td>CREATE STATISTICS Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1361</td>
</tr>
<tr>
<td>CREATE STATISTICS Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1366</td>
</tr>
<tr>
<td>CREATE TABLE Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1374</td>
</tr>
<tr>
<td>CREATE TABLE Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1379</td>
</tr>
<tr>
<td>DROP EXTENDED STORAGE Statement [Dynamic Tiering]</td>
<td>1392</td>
</tr>
<tr>
<td>DROP STATISTICS Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1393</td>
</tr>
<tr>
<td>DROP STATISTICS Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1395</td>
</tr>
<tr>
<td>EXPORT Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1399</td>
</tr>
<tr>
<td>EXPORT Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1403</td>
</tr>
<tr>
<td>GRANT EXTENDED STORAGE ADMIN Statement [Dynamic Tiering]</td>
<td>1407</td>
</tr>
<tr>
<td>IMPORT Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1409</td>
</tr>
<tr>
<td>IMPORT Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1414</td>
</tr>
<tr>
<td>IMPORT FROM Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1418</td>
</tr>
<tr>
<td>IMPORT FROM Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1424</td>
</tr>
<tr>
<td>INSERT Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1430</td>
</tr>
<tr>
<td>INSERT Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1433</td>
</tr>
<tr>
<td>MERGE DELTA Statement [Dynamic Tiering]</td>
<td>1435</td>
</tr>
<tr>
<td>REFRESH STATISTICS Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1437</td>
</tr>
<tr>
<td>REFRESH STATISTICS Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1441</td>
</tr>
<tr>
<td>UPDATE Statement (Extended Store Table) [Dynamic Tiering]</td>
<td>1445</td>
</tr>
<tr>
<td>UPDATE Statement (Multistore Table) [Dynamic Tiering]</td>
<td>1448</td>
</tr>
<tr>
<td>Additional Syntax [Dynamic Tiering]</td>
<td>1451</td>
</tr>
<tr>
<td>5.3 Streaming Analytics.</td>
<td>1452</td>
</tr>
<tr>
<td>ALTER DATABASE Statement [Streaming Analytics]</td>
<td>1452</td>
</tr>
<tr>
<td>ALTER SYSTEM INITIALIZE SERVICE Statement [Streaming Analytics]</td>
<td>1453</td>
</tr>
<tr>
<td>ALTER SYSTEM UNINITIALIZE SERVICE Statement [Streaming Analytics]</td>
<td>1454</td>
</tr>
<tr>
<td>5.4 Text Mining.</td>
<td>1455</td>
</tr>
<tr>
<td>Text Mining: TM_CATEGORIZE_KNN Function.</td>
<td>1456</td>
</tr>
</tbody>
</table>
6 System Views Reference .................................................... 1476
6.1 System Views ............................................................. 1476
  ABSTRACT_SQL_PLANS System View ........................................ 1477
  ACCESSIBLE_VIEWS System View ........................................... 1480
  ADAPTERS System View .................................................... 1480
  ADAPTER_CAPABILITIES System View ..................................... 1481
  ADAPTER_LOCATIONS System View ........................................ 1482
  AFL AREAS System View .................................................. 1483
  AFL_FUNCTIONS System View ............................................... 1484
  AFL_FUNCTION_PARAMETERS System View ............................... 1485
  AFL_FUNCTION_PROPERTIES System View ............................... 1486
  AFL_PACKAGES System View ............................................... 1487
  AFL_TEXTS System View .................................................. 1488
  AGENTS System View ..................................................... 1488
  AGENT_CONFIGURATION System View .................................... 1489
  AGENT_GROUPS System View ............................................... 1490
  ALL_AUDIT_LOG System View .............................................. 1490
  ANNOTATIONS System View ................................................ 1494
  ANONYMIZATION_VIEWS System View ................................... 1495
  ANONYMIZATION_VIEW_COLUMNS System View .......................... 1496
  APPLICATION_CONNECTION_HISTORY System View ...................... 1497
  APPLICATION_ENCRYPTION_KEYS System View .......................... 1498
  ASSOCIATIONS System View ............................................... 1499
  AUDIT_ACTIONS System View ............................................... 1500
  AUDIT_LOG System View ................................................... 1501
  AUDIT_POLICIES System View ............................................. 1503
  AUTHORIZATION_GRAPH System View ................................... 1506
  AUTHORIZATION_OBJECTS System View .................................. 1508
  AUTHORIZATION_TYPES System View .................................... 1509
  CDS_ANNOTATION_ASSIGNMENTS System View .......................... 1510
  CDS_ANNOTATION_VALUES System View .................................. 1511
  CDS_ARTIFACT_NAMES System View ...................................... 1512
  CDS_ASSOCIATIONS System View ......................................... 1513
  CDS_ENTITIES System View ............................................... 1514
  CDS_VIEWS System View .................................................. 1515
  CERTIFICATES System View ............................................... 1516
<table>
<thead>
<tr>
<th>System View</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENTSIDE_ENCRYPTION_COLUMN_KEYS</td>
<td>1517</td>
</tr>
<tr>
<td>CLIENTSIDE_ENCRYPTION_KEYPAIRS</td>
<td>1519</td>
</tr>
<tr>
<td>COLLATIONS System View</td>
<td>1520</td>
</tr>
<tr>
<td>COLUMNS System View - Deprecated</td>
<td>1520</td>
</tr>
<tr>
<td>CONFIGURATION_PARAMETER_PROPERTIES</td>
<td>1521</td>
</tr>
<tr>
<td>CONSTRAINTS System View</td>
<td>1523</td>
</tr>
<tr>
<td>CONVERT_INTERNAL_TYPE INTO_NAME System View</td>
<td>1524</td>
</tr>
<tr>
<td>CREDENTIALS System View</td>
<td>1524</td>
</tr>
<tr>
<td>CS_ALL_COLUMNS System View</td>
<td>1525</td>
</tr>
<tr>
<td>CS_BO_VIEWS System View</td>
<td>1526</td>
</tr>
<tr>
<td>CS_CONCAT_COLUMNS System View</td>
<td>1527</td>
</tr>
<tr>
<td>CS_FREESTYLE_COLUMNS System View</td>
<td>1528</td>
</tr>
<tr>
<td>CS_JOIN_CONDITIONS System View</td>
<td>1529</td>
</tr>
<tr>
<td>CS_JOIN_CONSTRAINTS System View</td>
<td>1530</td>
</tr>
<tr>
<td>CS_JOIN_PATHS System View</td>
<td>1531</td>
</tr>
<tr>
<td>CS_JOIN_TABLES System View</td>
<td>1532</td>
</tr>
<tr>
<td>CS_KEY FIGURES System View</td>
<td>1533</td>
</tr>
<tr>
<td>CS_VIEW_COLUMNS System View</td>
<td>1535</td>
</tr>
<tr>
<td>CS_VIEW_PARAMETERs System View</td>
<td>1536</td>
</tr>
<tr>
<td>DATA_STATISTICS System View</td>
<td>1538</td>
</tr>
<tr>
<td>DATA_TYPES System View</td>
<td>1542</td>
</tr>
<tr>
<td>DEPENDENCY RULES System View</td>
<td>1544</td>
</tr>
<tr>
<td>DEPENDENCY RULE_COLUMNs System View</td>
<td>1545</td>
</tr>
<tr>
<td>DYNAMIC RESULT_CACHE System View</td>
<td>1546</td>
</tr>
<tr>
<td>DYNAMIC RESULT_CACHE_INDEX_COLUMNS System View</td>
<td>1547</td>
</tr>
<tr>
<td>EFFECTIVE_APPLICATION_PRIVILEGES System View</td>
<td>1547</td>
</tr>
<tr>
<td>EFFECTIVE_MASK_EXPRESSIONS System View</td>
<td>1549</td>
</tr>
<tr>
<td>EFFECTIVE_PRIVILEGES System View</td>
<td>1550</td>
</tr>
<tr>
<td>EFFECTIVE_PRIVILEGE_GRANTEES System View</td>
<td>1552</td>
</tr>
<tr>
<td>EFFECTIVE_ROLES System View</td>
<td>1553</td>
</tr>
<tr>
<td>EFFECTIVE_ROLE_GRANTEES System View</td>
<td>1554</td>
</tr>
<tr>
<td>EFFECTIVE_STRUCTURED_PRIVILEGES System View</td>
<td>1555</td>
</tr>
<tr>
<td>ELEMENT_TYPES System View</td>
<td>1557</td>
</tr>
<tr>
<td>ENCRYPTION_ROOT_KEYS System View</td>
<td>1558</td>
</tr>
<tr>
<td>EPM_MODELS System View</td>
<td>1560</td>
</tr>
<tr>
<td>EPM_QUERY_SOURCES System View</td>
<td>1560</td>
</tr>
<tr>
<td>EXPLAIN_CALL_PLANS System View</td>
<td>1561</td>
</tr>
<tr>
<td>EXPLAIN_PLAN_TABLE System View</td>
<td>1565</td>
</tr>
<tr>
<td>FLEXIBLE_TABLES System View</td>
<td>1570</td>
</tr>
<tr>
<td>FULLTEXT_INDEXES System View</td>
<td>1571</td>
</tr>
<tr>
<td>FULL_SYSTEM_INFO_DUMPS System View</td>
<td>1574</td>
</tr>
</tbody>
</table>
PROJECTION_VIEW_COLUMN_SOURCES System View ............................. 1618
PSES System View ............................................................. 1619
PSE_CERTIFICATES System View ............................................ 1620
PSE_PUBLIC_KEYS System View ............................................. 1621
PSE_PURPOSE_OBJECTS System View ....................................... 1621
PUBLIC_KEYS System View .................................................. 1623
QUERY_PLANS System View .................................................. 1624
REFERENTIAL_CONSTRAINTS System View .................................. 1625
REMOTE_SOURCES System View ............................................. 1626
REMOTE_SOURCE_OBJECTS System View .................................... 1627
REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINTS System View ............ 1629
REMOTE_SOURCE_OBJECT_COLUMN_DESCRIPTIONS System View .......... 1630
REMOTE_SOURCE_OBJECT_DESCRIPTIONS System View ..................... 1631
REMOTE_SUBSCRIPTIONS System View ..................................... 1633
REMOTE_SUBSCRIPTION_DATA_CONTAINERS System View ................. 1635
REMOTE_SUBSCRIPTION_EXCEPTIONS System View ........................ 1636
REMOTE_USERS System View ................................................ 1637
REORG_GENERATE_OVERVIEW System View ................................ 1638
REORG_OVERVIEW System View ............................................. 1639
REORG_PLAN System View ................................................... 1640
REORG_PLAN_INFOS System View ........................................... 1641
REORG_STEPS System View .................................................. 1642
RESERVED_KEYWORDS System View ....................................... 1643
RESULT_CACHE System View .................................................. 1644
RESULT_CACHE_COLUMNS System View .................................... 1645
ROLES System View ............................................................ 1646
ROLE_LDAP_GROUPS System View ........................................... 1647
SAML_PROVIDERS System View ............................................. 1648
SAML_USER_MAPPINGS System View ......................................... 1649
SCHEDULER_JOB_PARAMETERS System View ................................ 1650
SCHEDULER_JOBS System View ............................................. 1650
SCHEMAS System View ...................................................... 1651
SEARCH_RULE_SETS System View ......................................... 1652
SEARCH_RULE_SET_CONDITIONS System View ............................... 1653
SEQUENCES System View .................................................... 1654
SERIES_KEY_COLUMNS System View .......................................... 1655
SERIES_TABLES System View .................................................. 1656
SESSION_COOKIES System View ............................................. 1658
SQLSCRIPT_ANALYZER_RULES System View ............................... 1659
SQLSCRIPT_VARIABLE_CACHE System view .................................. 1660
6.2 Monitoring Views ........................................................... 1729

M_ABSTRACT_SQL_PLAN_OVERVIEW System View ......................... 1730
M_ABSTRACT_SQL_PLAN_STATISTICS System View .......................... 1731
M_ACTIVE_PROCEDURES System View ........................................ 1733
M_ACTIVE_STATEMENTS System View ........................................ 1735
M_ADMISSION_CONTROL_EVENTS System View ............................ 1739
M_ADMISSION_CONTROL_QUEUES System View ......................... 1740
M_ADMISSION_CONTROL_STATISTICS System View ..................... 1741
M_AFL_FUNCTIONS System View ............................................. 1742
M_AFL_STATES System View ................................................ 1743
M_AGENTS System View .................................................... 1744
M_ANONYMIZATION_VIEWS System View .................................. 1745
MASYNCHRONOUS_TABLE_REPLICAS System View ...................... 1746
MASYNCHRONOUS_TABLE_REPLICAS_RESET System View .............. 1747
M_ATTACHED_STORAGES System View ...................................... 1747
M_BACKUP_CATALOG System View ......................................... 1748
M_BACKUP_CATALOG_FILES System View ................................ 1751
M_BACKUP_CONFIGURATION System View .................................. 1754
M_BACKUP_HISTORY_BROKEN System View ................................ 1756
M_BACKUP_PROGRESS System View ........................................ 1758
M_BACKUP_SIZE_ESTIMATIONS System View ............................... 1759
M_BLOCKED_TRANSACTIONS System View .................................. 1761
M_BUFFER_CACHE_POOL_STATISTICS System View ....................... 1762
M_BUFFER_CACHE_POOL_STATISTICS_RESET System View .............. 1764
M_BUFFER_CACHE_STATISTICS System View ................................ 1764
M_BUFFER_CACHE_STATISTICS_RESET System View ..................... 1765
M_CACHES System View ................................................... 1766
M_CACHES_RESET System View ............................................. 1767
M_CACHE_ENTRIES System View .......................................... 1768
M_CATALOG_MEMORY System View ........................................ 1769
M_CE_CALCSCECNARIOS System View ..................................... 1770
M_CE_CALCSCECNARIOS_OVERVIEW System View ....................... 1771
M_CE_CALSCENARIO_HINTS System View .................................. 1772
M_CE_CALSCVIEW_DEPENDENCIES System View .......................... 1772
M_CE_DEBUG_INFOS System View ......................................... 1773
M_CE_DEBUG_JSONS System View........................................... 1774
M_CE_DEBUG_NODE_MAPPING System View................................... 1775
M_CE_PLE_CALCSCENARIOS System View..................................... 1776
M_CLIENT_VERSIONS System View........................................... 1777
M_COLLECTION_TABLES System View........................................ 1778
M_COLLECTION_TABLE_VIRTUAL_FILES System View............................. 1779
M_COMPACTION_THREAD System View....................................... 1780
M_CONDITIONAL_VARIABLES System View..................................... 1782
M_CONDITIONAL_VARIABLES_RESET System View............................... 1784
M_CONFIGURATION_PARAMETER_VALUES System View........................... 1784
M_CONNECTIONS System View............................................. 1785
M_CONNECTION_STATISTICS System View..................................... 1790
M_CONSISTENCY_CHECK_HISTORY System View................................ 1795
M_CONSISTENCY_CHECK_HISTORY_ERRORS System View........................ 1797
M_CONTAINER_DIRECTORY System View...................................... 1799
M_CONTAINER_NAME_DIRECTORY System View................................. 1800
M_CONTEXT_MEMORY System View.......................................... 1801
M_CONTEXT_MEMORY_RESET System View.................................... 1804
M_CONVERTER_STATISTICS System View...................................... 1805
M_CONVERTER_STATISTICS_RESET System View................................ 1807
M_CS_ALL_COLUMNS System View.......................................... 1807
M_CS_ALL_COLUMN_STATISTICS System View................................ 1813
M_CS_COLUMNS System View.............................................. 1814
M_CS_COLUMNS_PERSISTENCE System View................................... 1818
M_CS_INDEXES System View............................................... 1819
M_CS_LOADS System View................................................ 1821
M_CS_LOB_SPACE_RECLAIMS System View.................................... 1822
M_CS_LOG_REPLAY_QUEUE_STATISTICS System View............................ 1824
M_CS_LOG_REPLAY_QUEUE_STATISTICS_RESET System View....................... 1825
M_CS_MVCC System View................................................. 1826
M_CS_NSE_ADVISOR System View........................................... 1829
M_CS_NSE_ADVISORDETAILS System View.................................. 1830
M_CS_NSE_ADVISOR_STATISTICS System View................................ 1832
M_CS_PARTITIONSSystem View - Deprecated................................ 1833
M_CS_RECORD_LOCKSTATISTICS System View................................... 1834
M_CS_TABLE_HANDLES System View......................................... 1835
M_CS_TABLE_LOCKSSystem View........................................... 1836
M_CS_TABLES System View................................................ 1837
M_CS_UNLOADS System View............................................... 1842
M_CUSTOMIZABLE_FUNCTIONALITIES System View............................ 1844
M_DATABASE System View................................................. 1844
<table>
<thead>
<tr>
<th>System View</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_DATABASES System View</td>
<td>1846</td>
</tr>
<tr>
<td>M_DATABASE_HISTORY System View</td>
<td>1847</td>
</tr>
<tr>
<td>M_DATABASE_REPLICAS System View</td>
<td>1848</td>
</tr>
<tr>
<td>M_DATABASE_REPLICA_STATISTICS System View</td>
<td>1849</td>
</tr>
<tr>
<td>M_DATA_STATISTICS System View</td>
<td>1851</td>
</tr>
<tr>
<td>M_DATA_VOLUMES System View</td>
<td>1854</td>
</tr>
<tr>
<td>M_DATA_VOLUME_PAGE_STATISTICS System View</td>
<td>1855</td>
</tr>
<tr>
<td>M_DATA_VOLUME_PAGE_STATISTICS_RESET System View</td>
<td>1857</td>
</tr>
<tr>
<td>M_DATA_VOLUME_PARTITION_STATISTICS System View</td>
<td>1858</td>
</tr>
<tr>
<td>M_DATA_VOLUME_RECLAIM_STATISTICS System View</td>
<td>1859</td>
</tr>
<tr>
<td>M_DATA_VOLUME_STATISTICS System View</td>
<td>1861</td>
</tr>
<tr>
<td>M_DATA_VOLUME_SUPERBLOCK_STATISTICS System View</td>
<td>1862</td>
</tr>
<tr>
<td>M_DEBUG_CONNECTIONS System View</td>
<td>1864</td>
</tr>
<tr>
<td>M_DEBUG_SESSIONS System View</td>
<td>1865</td>
</tr>
<tr>
<td>M_DELTA_MERGE_STATISTICS System View</td>
<td>1866</td>
</tr>
<tr>
<td>M_DISK_USAGE System View</td>
<td>1871</td>
</tr>
<tr>
<td>M_DSO_OPERATIONS System View</td>
<td>1872</td>
</tr>
<tr>
<td>M_DYNAMIC_RESULT_CACHE System View</td>
<td>1873</td>
</tr>
<tr>
<td>M_DYNAMIC_RESULT_CACHE_EXCLUSIONS System View</td>
<td>1875</td>
</tr>
<tr>
<td>M_EFFECTIVE_PASSWORD_POLICY System View</td>
<td>1877</td>
</tr>
<tr>
<td>M_EFFECTIVE_TABLE_PLACEMENT System View</td>
<td>1879</td>
</tr>
<tr>
<td>M_ENCRYPTION_OVERVIEW System View</td>
<td>1880</td>
</tr>
<tr>
<td>M_EPM_SESSIONS System View</td>
<td>1881</td>
</tr>
<tr>
<td>M_ERROR_CODES System View</td>
<td>1882</td>
</tr>
<tr>
<td>M_ES_CONNECTIONS System View</td>
<td>1883</td>
</tr>
<tr>
<td>M_ES_DBSPACES System View</td>
<td>1883</td>
</tr>
<tr>
<td>M_ES_DBSPACE_FILES System View</td>
<td>1885</td>
</tr>
<tr>
<td>M_ES_DELTA_MEMORY System View</td>
<td>1886</td>
</tr>
<tr>
<td>M_ES_DELTA_MERGE_STATISTICS System View</td>
<td>1888</td>
</tr>
<tr>
<td>M_ES_DELTA_MERGE_STATISTICS System View</td>
<td>1889</td>
</tr>
<tr>
<td>M_ES_EXPENSIVE_STATEMENTS System View</td>
<td>1890</td>
</tr>
<tr>
<td>M_ES_LOCKS System View</td>
<td>1891</td>
</tr>
<tr>
<td>M_ES_OVERVIEW System View</td>
<td>1893</td>
</tr>
<tr>
<td>M_ES_RESULT_CACHE System View</td>
<td>1894</td>
</tr>
<tr>
<td>M_ES_SERVICE_REPLICATION System View</td>
<td>1895</td>
</tr>
<tr>
<td>M_ES_TABLES System View</td>
<td>1897</td>
</tr>
<tr>
<td>M_ES_TRANSACTIONS System View</td>
<td>1899</td>
</tr>
<tr>
<td>M_EVENTS System View</td>
<td>1901</td>
</tr>
<tr>
<td>M_EXECUTED_STATEMENTS System View</td>
<td>1903</td>
</tr>
<tr>
<td>M_EXPENSIVE_STATEMENTS System View</td>
<td>1905</td>
</tr>
<tr>
<td>M_EXPENSIVE_STATEMENT_EXECUTION_LOCATION_STATISTICS System View</td>
<td>1910</td>
</tr>
<tr>
<td>View Operation</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------</td>
</tr>
<tr>
<td>M_LIVECACHE_LOCK_STATISTICS</td>
<td>1972</td>
</tr>
<tr>
<td>M_LIVECACHE_LOCK_STATISTICS_RESET</td>
<td>1973</td>
</tr>
<tr>
<td>M_LIVECACHE_OMS_VERSIONS</td>
<td>1974</td>
</tr>
<tr>
<td>M_LIVECACHE_PROCEDURE_STATISTICS</td>
<td>1975</td>
</tr>
<tr>
<td>M_LIVECACHE_PROCEDURE_STATISTICS_RESET</td>
<td>1986</td>
</tr>
<tr>
<td>M_LIVECACHE_SCHEMA_STATISTICS</td>
<td>1986</td>
</tr>
<tr>
<td>M_LIVECACHE_SCHEMA_STATISTICS_RESET</td>
<td>1987</td>
</tr>
<tr>
<td>M_LOAD_HISTORY_HOST</td>
<td>1987</td>
</tr>
<tr>
<td>M_LOAD_HISTORY_INFO</td>
<td>1989</td>
</tr>
<tr>
<td>M_LOAD_HISTORY_SERVICE</td>
<td>1990</td>
</tr>
<tr>
<td>M_LOCK_WAITS_STATISTICS</td>
<td>1994</td>
</tr>
<tr>
<td>M_LOG_BUFFERS</td>
<td>1995</td>
</tr>
<tr>
<td>M_LOG_BUFFERS_RESET</td>
<td>1996</td>
</tr>
<tr>
<td>M_LOG_PARTITION</td>
<td>1997</td>
</tr>
<tr>
<td>M_LOG_PARTITION_RESET</td>
<td>2001</td>
</tr>
<tr>
<td>M_LOG_REPLAY_QUEUE_STATISTICS</td>
<td>2002</td>
</tr>
<tr>
<td>M_LOG_REPLAY_QUEUE_STATISTICS_RESET</td>
<td>2003</td>
</tr>
<tr>
<td>M_LOG_SEGMENTS</td>
<td>2003</td>
</tr>
<tr>
<td>M_LOG_SEGMENTS_RESET</td>
<td>2006</td>
</tr>
<tr>
<td>M_MEMORY</td>
<td>2007</td>
</tr>
<tr>
<td>M_MEMORY_OBJECTS</td>
<td>2008</td>
</tr>
<tr>
<td>M_MEMORY_OBJECTS_RESET</td>
<td>2010</td>
</tr>
<tr>
<td>M_MEMORY_OBJECT_DISPOSITIONS</td>
<td>2011</td>
</tr>
<tr>
<td>M_MEMORY_PROFILER</td>
<td>2013</td>
</tr>
<tr>
<td>M_MEMORY_RECLAIM_STATISTICS</td>
<td>2014</td>
</tr>
<tr>
<td>M_MEMORY_RECLAIM_STATISTICS_RESET</td>
<td>2017</td>
</tr>
<tr>
<td>M_MERGED_TRACES</td>
<td>2017</td>
</tr>
<tr>
<td>M_METADATA_CACHE_STATISTICS</td>
<td>2019</td>
</tr>
<tr>
<td>M_MONITORS</td>
<td>2020</td>
</tr>
<tr>
<td>M_MONITOR_COLUMNS</td>
<td>2021</td>
</tr>
<tr>
<td>M_MULTIDIMENSIONAL_STATEMENT_STATISTICS</td>
<td>2023</td>
</tr>
<tr>
<td>M_MUTEXES</td>
<td>2025</td>
</tr>
<tr>
<td>M_MUTEXES_RESET</td>
<td>2027</td>
</tr>
<tr>
<td>M_MVCC_OVERVIEW</td>
<td>2027</td>
</tr>
<tr>
<td>M_MVCC_SNAPSHOTS</td>
<td>2029</td>
</tr>
<tr>
<td>M_MVCC_TABLES</td>
<td>2030</td>
</tr>
<tr>
<td>M_NUMA_NODES</td>
<td>2030</td>
</tr>
<tr>
<td>M_NUMA_RESOURCES</td>
<td>2032</td>
</tr>
<tr>
<td>M_OBJECT_LOCKS</td>
<td>2033</td>
</tr>
<tr>
<td>M_OBJECT_LOCK_STATISTICS</td>
<td>2035</td>
</tr>
<tr>
<td>System View</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>M_OUT_OF_MEMORY_EVENTS System View</td>
<td>2035</td>
</tr>
<tr>
<td>M_OUT_OF_MEMORY_EVENTS_RESET System View</td>
<td>2036</td>
</tr>
<tr>
<td>M_PAGEACCESS_STATISTICS System View</td>
<td>2037</td>
</tr>
<tr>
<td>M_PAGEACCESS_STATISTICS_RESET System View</td>
<td>2038</td>
</tr>
<tr>
<td>M_PASSWORD_POLICY System View</td>
<td>2039</td>
</tr>
<tr>
<td>M_PERFTRACE System View</td>
<td>2041</td>
</tr>
<tr>
<td>M_PERSISTENCE_ENCRYPTION.Keys System View</td>
<td>2042</td>
</tr>
<tr>
<td>M_PERSISTENCE_ENCRYPTION.STATUS System View</td>
<td>2043</td>
</tr>
<tr>
<td>M_PERSISTENCE_MANAGERS System View</td>
<td>2044</td>
</tr>
<tr>
<td>M_PERSISTENCE_MANAGERS_RESET System View</td>
<td>2046</td>
</tr>
<tr>
<td>M_PERSISTENT_MEMORY_VOLUMES System View</td>
<td>2047</td>
</tr>
<tr>
<td>M_PERSISTENT_MEMORY_VOLUME_DATA_FILES System View</td>
<td>2048</td>
</tr>
<tr>
<td>M_PERSISTENT_MEMORY_VOLUME_STATISTICS System View</td>
<td>2049</td>
</tr>
<tr>
<td>M_PERSISTENT_MEMORY_VOLUME_STATISTICS_RESET System View</td>
<td>2051</td>
</tr>
<tr>
<td>M_PLE_RUNTIME_OBJECTS System View</td>
<td>2052</td>
</tr>
<tr>
<td>M_PLE_SESSIONS System View</td>
<td>2053</td>
</tr>
<tr>
<td>M_PLUGIN_MANIFESTS System View</td>
<td>2054</td>
</tr>
<tr>
<td>M_PLUGIN_STATUS System View</td>
<td>2054</td>
</tr>
<tr>
<td>M_PREPARED_STATEMENTS System View</td>
<td>2055</td>
</tr>
<tr>
<td>M_QUERY_CACHED_PLANS System View - Deprecated</td>
<td>2059</td>
</tr>
<tr>
<td>M_READWRITELOCKS System View</td>
<td>2060</td>
</tr>
<tr>
<td>M_READWRITELOCKS_RESET System View</td>
<td>2063</td>
</tr>
<tr>
<td>M_RECORD_LOCKS System View</td>
<td>2063</td>
</tr>
<tr>
<td>M_RECOVERY_PROGRESS System View</td>
<td>2064</td>
</tr>
<tr>
<td>M_REMOTE_CONNECTIONS System View</td>
<td>2066</td>
</tr>
<tr>
<td>M_REMOTE_SOURCES System View</td>
<td>2067</td>
</tr>
<tr>
<td>M_REMOTE_SOURCE_LATENCY_HISTORY System View</td>
<td>2068</td>
</tr>
<tr>
<td>M_REMOTE_SOURCE_LATENCY_STATUS System View</td>
<td>2068</td>
</tr>
<tr>
<td>M_REMOTE_SOURCE_STATISTICS System View</td>
<td>2069</td>
</tr>
<tr>
<td>M_REMOTE_STATEMENTS System View</td>
<td>2069</td>
</tr>
<tr>
<td>M_REMOTE_SUBSCRIPTIONS System View</td>
<td>2070</td>
</tr>
<tr>
<td>M_REMOTE_SUBSCRIPTION_COMMIT_SEQUENCE_CONTAINERS System View</td>
<td>2072</td>
</tr>
<tr>
<td>M_REMOTE_SUBSCRIPTION_COMMIT_SEQUENCE_GROUP_CONTAINERS System View</td>
<td>2074</td>
</tr>
<tr>
<td>M_REMOTE_SUBSCRIPTION_COMPONENTS System View</td>
<td>2075</td>
</tr>
<tr>
<td>M_REMOTE_SUBSCRIPTION_DISTRIBUTOR_QUEUE_DATA_CONTAINERS System View</td>
<td>2075</td>
</tr>
<tr>
<td>M_REMOTE_SUBSCRIPTION_LOB_CONTAINERS System View</td>
<td>2076</td>
</tr>
<tr>
<td>M_REMOTE_SUBSCRIPTION_ROLLBACK_SAVEPOINT_CONTAINERS System View</td>
<td>2077</td>
</tr>
<tr>
<td>M_REMOTE_SUBSCRIPTION_STATISTICS System View</td>
<td>2078</td>
</tr>
<tr>
<td>M_REMOTE_SUBSCRIPTION_TRANSACTION_CONTAINERS System View</td>
<td>2079</td>
</tr>
<tr>
<td>M_REMOTE_TABLE_REPLICAS System View</td>
<td>2081</td>
</tr>
<tr>
<td>M_REORG_ALGORITHMS System View</td>
<td>2083</td>
</tr>
</tbody>
</table>

SAP HANA SQL Reference Guide for SAP HANA Platform
<table>
<thead>
<tr>
<th>System View</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_REPLICATION_LOG System View</td>
<td>2083</td>
</tr>
<tr>
<td>M_REPO_TRANSPORT_FILES System View</td>
<td>2084</td>
</tr>
<tr>
<td>M_RESULT_CACHE System View</td>
<td>2084</td>
</tr>
<tr>
<td>M_RESULT_CACHE_EXCLUSIONS System View</td>
<td>2086</td>
</tr>
<tr>
<td>M_RESULT_CACHE_RESET System View</td>
<td>2086</td>
</tr>
<tr>
<td>M_RS_INDEXES System View</td>
<td>2087</td>
</tr>
<tr>
<td>M_RS_MEMORY System View</td>
<td>2089</td>
</tr>
<tr>
<td>M_RS_TABLES System View</td>
<td>2089</td>
</tr>
<tr>
<td>M_RS_TABLE_VERSION_STATISTICS System View</td>
<td>2091</td>
</tr>
<tr>
<td>M_SAVEPOINTS System View</td>
<td>2093</td>
</tr>
<tr>
<td>M_SAVEPOINT_STATISTICS System View</td>
<td>2097</td>
</tr>
<tr>
<td>M_SAVEPOINT_STATISTICS_RESET System View</td>
<td>2101</td>
</tr>
<tr>
<td>M_SCHEDULER_JOBS System View</td>
<td>2101</td>
</tr>
<tr>
<td>M_SECURESTORE System View</td>
<td>2102</td>
</tr>
<tr>
<td>M_SEMAPHORES System View</td>
<td>2103</td>
</tr>
<tr>
<td>M_SEMAPHORES_RESET System View</td>
<td>2104</td>
</tr>
<tr>
<td>M_SEQUENCES System View</td>
<td>2105</td>
</tr>
<tr>
<td>M_SERIES_TABLES System View</td>
<td>2106</td>
</tr>
<tr>
<td>M_SERVICES System View</td>
<td>2107</td>
</tr>
<tr>
<td>M_SERVICE_COMPONENTDETAILS System View</td>
<td>2108</td>
</tr>
<tr>
<td>M_SERVICE_COMPONENT_HISTORY System View</td>
<td>2109</td>
</tr>
<tr>
<td>M_SERVICE_COMPONENT_MEMORY System View</td>
<td>2110</td>
</tr>
<tr>
<td>M_SERVICE_COMPONENTS System View</td>
<td>2111</td>
</tr>
<tr>
<td>M_SERVICE_MEMORY System View</td>
<td>2113</td>
</tr>
<tr>
<td>M_SERVICE_NETWORK_IO System View</td>
<td>2115</td>
</tr>
<tr>
<td>M_SERVICE_NETWORK_IO_RESET System View</td>
<td>2116</td>
</tr>
<tr>
<td>M_SERVICE_NETWORK_METHOD_IO System View</td>
<td>2117</td>
</tr>
<tr>
<td>M_SERVICE_NETWORK_METHOD_IO_RESET System View</td>
<td>2118</td>
</tr>
<tr>
<td>M_SERVICE_REPLICATION System View</td>
<td>2118</td>
</tr>
<tr>
<td>M_SERVICE_STATISTICS System View</td>
<td>2122</td>
</tr>
<tr>
<td>M_SERVICE_THREADS System View</td>
<td>2124</td>
</tr>
<tr>
<td>M_SERVICE_THREAD_CALLSTACKS System View</td>
<td>2128</td>
</tr>
<tr>
<td>M_SERVICE_THREAD_SAMPLES System View</td>
<td>2129</td>
</tr>
<tr>
<td>M_SERVICE_TRACES System View</td>
<td>2132</td>
</tr>
<tr>
<td>M_SERVICE_TYPES System View</td>
<td>2132</td>
</tr>
<tr>
<td>M_SESSION_CONTEXT System View</td>
<td>2133</td>
</tr>
<tr>
<td>M_SHARED_MEMORY System View</td>
<td>2134</td>
</tr>
<tr>
<td>M_SNAPSHOTS System View</td>
<td>2134</td>
</tr>
<tr>
<td>M_SQLSCRIPT_CODE_COVERAGEOBJECT_DEFINITIONS System View</td>
<td>2135</td>
</tr>
<tr>
<td>M_SQLSCRIPT_CODE_COVERAGE_RESULTS System View</td>
<td>2136</td>
</tr>
<tr>
<td>M_SQLSCRIPT_PLAN_PROFILERS System View</td>
<td>2137</td>
</tr>
<tr>
<td>System View</td>
<td>Page</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>M_SQLSCRIPT_PLAN_PROFILER_RESULTS</td>
<td>2139</td>
</tr>
<tr>
<td>M_SQLSCRIPT_VARIABLE_CACHE</td>
<td>2142</td>
</tr>
<tr>
<td>M_SQL_ANALYZER_PLAN_TRACES</td>
<td>2143</td>
</tr>
<tr>
<td>M_SQL_CLIENT_NETWORK_IO</td>
<td>2144</td>
</tr>
<tr>
<td>M_SQL_PLAN_CACHE</td>
<td>2145</td>
</tr>
<tr>
<td>M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS</td>
<td>2155</td>
</tr>
<tr>
<td>M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS_RESET</td>
<td>2157</td>
</tr>
<tr>
<td>M_SQL_PLAN_CACHE_OVERVIEW</td>
<td>2157</td>
</tr>
<tr>
<td>M_SQL_PLAN_CACHE_PARAMETERS</td>
<td>2157</td>
</tr>
<tr>
<td>M_SQL_PLAN_CACHE_RESET</td>
<td>2160</td>
</tr>
<tr>
<td>M_SQL_PLAN_CACHE_PARAMETERS</td>
<td>2161</td>
</tr>
<tr>
<td>M_SQL_PLAN_STATISTICS</td>
<td>2161</td>
</tr>
<tr>
<td>M_SQL_PLAN_STATISTICS_RESET</td>
<td>2169</td>
</tr>
<tr>
<td>M_STATISTICS_LASTVALUES</td>
<td>2170</td>
</tr>
<tr>
<td>M_STREAMING_APPLICATIONS</td>
<td>2171</td>
</tr>
<tr>
<td>M_STREAMING_PROJECTS</td>
<td>2172</td>
</tr>
<tr>
<td>M_STREAMING_PROJECT_ADAPTERS</td>
<td>2174</td>
</tr>
<tr>
<td>M_STREAMING_PROJECT_ADAPTER_STATISTICS</td>
<td>2176</td>
</tr>
<tr>
<td>M_STREAMING_PROJECT_GD_SUBSCRIPTIONS</td>
<td>2177</td>
</tr>
<tr>
<td>M_STREAMING_PROJECT_PUBLISHERS</td>
<td>2178</td>
</tr>
<tr>
<td>M_STREAMING_PROJECT_STREAMS</td>
<td>2180</td>
</tr>
<tr>
<td>M_STREAMING_PROJECT_SUBSCRIBERS</td>
<td>2182</td>
</tr>
<tr>
<td>M_STREAMING_SCHEMAS</td>
<td>2185</td>
</tr>
<tr>
<td>M_STREAMING_SERVICES</td>
<td>2186</td>
</tr>
<tr>
<td>M_SYSTEM_AVAILABILITY</td>
<td>2187</td>
</tr>
<tr>
<td>M_SYSTEM_DATA_STATISTICS</td>
<td>2189</td>
</tr>
<tr>
<td>M_SYSTEM_INFORMATION_STATEMENTS</td>
<td>2192</td>
</tr>
<tr>
<td>M_SYSTEM_LIMITS</td>
<td>2192</td>
</tr>
<tr>
<td>M_SYSTEM_OVERVIEW</td>
<td>2193</td>
</tr>
<tr>
<td>M_SYSTEM_REPLICATION</td>
<td>2194</td>
</tr>
<tr>
<td>M_SYSTEM_REPLICATION_MVCC_HISTORY</td>
<td>2196</td>
</tr>
<tr>
<td>M_SYSTEM_REPLICATION_TAKEOVER_HISTORY</td>
<td>2197</td>
</tr>
<tr>
<td>M_SYSTEM_REPLICATION_TIMETRAVEL</td>
<td>2199</td>
</tr>
<tr>
<td>M_TABLES</td>
<td>2200</td>
</tr>
<tr>
<td>M_TABLE_LOB_FILES</td>
<td>2201</td>
</tr>
<tr>
<td>M_TABLE_LOB_STATISTICS</td>
<td>2202</td>
</tr>
<tr>
<td>M_TABLE_LOCATIONS</td>
<td>2204</td>
</tr>
<tr>
<td>M_TABLE_LOCKS</td>
<td>2206</td>
</tr>
<tr>
<td>M_TABLE_PARTITION_STATISTICS</td>
<td>2207</td>
</tr>
<tr>
<td>M_TABLE_PARTITIONS</td>
<td>2209</td>
</tr>
<tr>
<td>M_TABLE_PERSISTENCE_LOCATIONS</td>
<td>2212</td>
</tr>
<tr>
<td>M_TABLE_PERSISTENCE_LOCATION_STATISTICS</td>
<td>2213</td>
</tr>
</tbody>
</table>
7 System Views Reference for Additional SAP HANA Contexts. ........................................ 2296

7.1 Dynamic Tiering. ........................................................................................................ 2296
M_ES_CONNECTIONS System View [Dynamic Tiering]. ........................................... 2297
M_ES_DBSPACE_FILES System View [Dynamic Tiering]. ................................... 2298
M_ES_DBSPACES System View [Dynamic Tiering]. ........................................... 2299
M_ES_DELTA_MEMORY System View [Dynamic Tiering]. ................................... 2300
M_ES_DELTA_MERGE_STATISTICS System View [Dynamic Tiering]. ................. 2301
M_ES_EXPENSIVE_STATEMENTS System View [Dynamic Tiering]. ......................... 2302
M_ES_LOCKS System View [Dynamic Tiering]. .................................................. 2302
M_ES_OVERVIEW System View [Dynamic Tiering]. ........................................... 2303
M_ES_RESULT_CACHE [Dynamic Tiering]. .................................................... 2304
M_ES_TABLES System View [Dynamic Tiering]. .................................................. 2305
M_ES_TRANSACTIONS System View [Dynamic Tiering]. ................................... 2306
Additional System Views [Dynamic Tiering]. ..................................................... 2308

7.2 Streaming Analytics. ............................................................................................... 2309
M_STREAMING_APPLICATIONS System View [Streaming Analytics]. ................. 2310
M_STREAMING_PROJECTS System View [Streaming Analytics]. ......................... 2311
M_STREAMING_PROJECT_ADAPTERS System View [Streaming Analytics]. ......... 2314
M_STREAMING_PROJECT_ADAPTER_STATISTICS System View [Streaming Analytics]. . 2316
M_STREAMING_PROJECT_GD_SUBSCRIPTIONS System View [Streaming Analytics]. . 2317
M_STREAMING_PROJECT_PUBLISHERS System View [Streaming Analytics]. ......... 2318
M_STREAMING_PROJECT_SUBSCRIBERS System View [Streaming Analytics]. ......... 2321
M_STREAMING_PROJECT_STREAMS System View [Streaming Analytics]. ............. 2324
M_STREAMING_SCHEMAS System View [Streaming Analytics]. ......................... 2326
M_STREAMING_SERVICES System View [Streaming Analytics]. ......................... 2327
Additional Views For Streaming Analytics. ........................................................... 2327
1 SAP HANA SQL Reference Guide for SAP HANA Platform

The SAP HANA SQL Reference describes all SQL data types, predicates, operators, expressions, functions, statements, and error codes. The SAP HANA System Views Reference describes all SYS schema views that allow you to query for various information about the system state using SQL operations.
2 SAP HANA SQL Reference Guide (New and Changed)

SAP HANA Platform 2.0 SPS 07 introduces new and changed features for the SAP HANA SQL and System Views Reference

SQL Statements (New and Changed)

**ALTER SYSTEM {ADD | ALTER | REMOVE} STATEMENT HINT Statement (Changed)**
You can now specify a procedure/function definition for the target query. When specified, the given hint is implicitly applied. ALTER SYSTEM {ADD | ALTER | REMOVE} STATEMENT HINT Statement (System Management) [page 493]

**ALTER SYSTEM ALTER AUDIT LOG (System Management) (New)**
- The audit log table is now configured to use SAP HANA Native Storage Extensions (NSE) by default. You can also choose to load it completely into memory as before. ALTER SYSTEM ALTER AUDIT LOG (System Management) [page 499]
- The `<load_unit>` value has been extended to include COLUMN. ALTER SYSTEM ALTER AUDIT LOG (System Management) [page 499]

**ALTER SYSTEM RELOAD FILE PSE Statement (New)**
This SQL statement enforces changes to PSE stores without requiring a database restart. ALTER SYSTEM RELOAD FILE PSE Statement (System Management) [page 561]

**BACKUP Statements (Changed)**
It is now possible to flag complete data backups to be retained. If backups are flagged as retained, they cannot be deleted by the BACKUP CATALOG DELETE command or by the scheduled housekeeping tasks.

**CALL Statement (Changed)**
The CALL statement now supports the cascading of hints into internal SQL statements after optimization. When there are nested CALL statements, CASCADE hints are also propagated into the nested procedures. CALL Statement (Procedural) [page 715]

**CREATE/ALTER JWT and CREATE/ALTER SAML PROVIDER Statements (Changed)**
Automatic user creation is now supported for SAML and JWT with LDAP authorization. The JWT and SAML system views have been update to reflect this information. CREATE JWT PROVIDER Statement (Access Control) [page 767], ALTER JWT PROVIDER Statement (Access Control) [page 455], CREATE SAML PROVIDER Statement (Access Control) [page 803], ALTER SAML PROVIDER Statement (Access Control) [page 476], JWT_PROVIDERS System View [page 1589], SAML_PROVIDERS System View [page 1648]

**CREATE/ALTER WORKLOAD MAPPINGS Statement (Changed)**
You can now include APPLICATION SOURCE as a key value when creating and altering workload mappings. The WORKLOAD_MAPPINGS System View has also been extended to display this value. CREATE
CREATE AUDIT POLICY Statement (Changed)

- Audit policies now support the specification of user groups in addition to users for explicit inclusion or exclusion in the logging of audit events. The AUDIT_POLICIES System View has been modified to reflect this change.
- You can now specify schema names for some DDL based audit actions. The Audit Actions system view has been extended to include this information.
- The audit actions IMPORT, IMPORT SCAN, and EXPORT statements can now be tracked by the audit policy.
- CREATE AUDIT POLICY Statement (Access Control) [page 722], AUDIT_POLICIES System View [page 1503], AUDIT_ACTIONS System View [page 1500]

CREATE CERTIFICATE Statement (Changed)

Duplicate certificates are supported as long as a certificate name is specified and unique. CREATE CERTIFICATE Statement (System Management) [page 739]

CREATE FUZZY SEARCH INDEX Statement (New) This new statement is an alternative to using the CREATE FULLTEXT INDEX Statement to create a fulltext index on a specified table column. CREATE FUZZY SEARCH INDEX Statement (Data Definition) [page 759]

CREATE WORKLOAD CLASS Statement (Changed)

Some limitations to support thread limits on hierarchical workload class have been removed. CREATE WORKLOAD CLASS Statement (Workload Management) [page 929]

GRANT Statement (Changed) The PARTITION ADMIN system privilege has been introduced to allow the execution of non-destructive partitioning clauses in the ALTER TABLE Statement. GRANT Statement (Access Control) [page 1010], Non-heterogeneous Alter Partition Clauses [page 630], Heterogeneous Alter Partition Clauses [page 640].

SELECT Statement (Changed)

- The SELECT statement now supports the parameters WAIT, NOWAIT, and IGNORE LOCKED in the FOR SHARE LOCK clause.
- A new clause, `<collation_clause>` supports collation at the statement.
- SELECT Statement (Data Manipulation) [page 1104]

SQL Functions (New and Changed)

ADD_WORKDAYS and WORKDAYS_BETWEEN Function (Changed)

- These functions now support the Factory And Holiday Calendar.
- The parameters `<table_name_suffix>` and `<client>` have been added to these functions.
- ADD_WORKDAYS Function (Datetime) [page 98], WORKDAYS_BETWEEN Function (Datetime) [page 400]

SCORE Function (Miscellaneous) (Changed) The SCORE function has been extended to support new clauses. SCORE Function (Fulltext) [page 314]

GROUP_SCORE Function (Miscellaneous) (New) This new function returns the relevance of grouped records found. GROUP_SCORE Function (Miscellaneous) [page 199]

FIRST_VALUE Function (Aggregate) (Changed)
This function now returns spatial data types (ST_Geometry, ST_Point) when it is executed with spatial types as input parameter. FIRST_VALUE Function (Aggregate) [page 195]

LAST_VALUE Function (Aggregate) (Changed)

This function now returns spatial data types (ST_Geometry, ST_Point) when it is executed with spatial types as input parameter. LAST_VALUE Function (Aggregate) [page 238]

NTH_VALUE Function (Aggregate) (Changed)

This function now returns spatial data types (ST_Geometry, ST_Point) when it is executed with spatial types as input parameter. NTH_VALUE Function (Aggregate) [page 279]

System Views (New and Changed)

AUDIT ACTIONS System View (Changed) The system view now includes information on whether the specified audit action can be specified for a specific schema. AUDIT_ACTIONS System View [page 1500]

M_CACHE and M_CACHE_ENTRIES System Views (Changed) These system views now include information on the time of the last access of the cache instance. M_CACHES System View [page 1766], M_CACHE_ENTRIES System View [page 1768]

M_CS_TABLE_LOCKS and M_CS_TABLE_HANDLES System Views (New) Two public monitoring views are now available to provide information about locked tables and threads waiting to lock tables, from the perspective of the Column Store lock manager (IndexMgr). M_CS_TABLE_HANDLES System View [page 1835], M_CS_TABLE_LOCKS System View [page 1836]

M_DATA_VOLUME_RECLAIM_STATISTICS System View (New) A new view has been introduced to display reclamation statistics on a data volume. M_DATA_VOLUME_RECLAIM_STATISTICS System View [page 1859]

M_PREPARED_STATEMENTS and M_ACTIVE_STATEMENTS System Views (Changed)

The M_PREPARED_STATEMENTS and M_ACTIVE_STATEMENTS views have been enhanced to support thread limits. M_PREPARED_STATEMENTS System View [page 2055], M_ACTIVE_STATEMENTS System View [page 1735]

M_REMOTE_STATEMENTS and M_REMOTE_CONNECTIONS System Views (Changed)

The REMOTE_DURATION column has been added to this view. M_REMOTE_STATEMENTS System View [page 2069]

M_TABLE_PARTITIONS System View (Changed) The column PERSISTENT_MEMORY_SIZE_IN_TOTAL has been added to the view to display persistent memory consumption (in bytes) for every column store table. M_TABLE_PARTITIONS System View [page 2209]

M_SQL_PLAN_CACHE System Views (Changed)

A new value has been added to the COMPILATION_OPTIONS column of this view to return a STATEMENT_HINT_ON INNER STATEMENT option when the SQLScript compiler applies a statement hint to one of the inner statements of this SQLScript plan. M_SQL_PLAN_CACHE System View [page 2145]

STATEMENT_HINTS System Views (Changed)

Three new columns have been added to this view to provide additional information on statement hints. STATEMENT_HINTS System View [page 1661]

WORKLOAD MAPPINGS System View (Changed)

- The IS_VALID column has been added to this view to indicate whether the workload mapping is valid or not.
• The columns APPLICATION_SOURCE and APPLICATION_SOURCE_WILDCARD have been added to the view.
• WORKLOAD_MAPPINGS System View [page 1723]

Embedded Statistics System View (New and Changed)

Other Changes by Feature Name (New and Changed)
3 Introduction

This reference guide describes the syntax and semantics of SAP HANA SQL statements, built-in functions, and SYS schema system views in the SAP HANA database system. This document applies to all supported platforms in the same manner unless mentioned otherwise.

SAP HANA server software and tools are used in many SAP HANA platform and installation scenarios. The feature capability of the SAP HANA server can depend on the type of SAP HANA license as well as any additional capabilities that have been installed separately. Refer to the Feature Scope Description for SAP HANA for your specific SAP HANA version located on the SAP HANA Platform webpage for information about the capabilities available for your license and installation scenario.

⚠️ Caution

This guide contains syntax variations for different product contexts. These are handled as separate reference topics that have titles that announce the context in square brackets at the end of the title. The guide is also organized to keep reference topics together for a context. Always be sure you are using the SQL reference topic specific to your context.
4 SQL Reference

Introduction to SQL [page 27]
This chapter describes the SAP HANA database implementation of Structured Query Language (SQL).

SQL Notation Conventions [page 30]
SQL syntax notation conventions used in this guide.

Data Types [page 33]
A data type defines the characteristics of a data value. A special value of NULL is included in every data type to indicate the absence of a value.

Reserved Words [page 50]
Reserved words are words which have a special meaning to the SQL parser in the SAP HANA database and cannot be used as when defining an identifier.

Operators [page 51]
Use operators to perform arithmetic operations in expressions. Operators can be used for calculation, value comparison, or to assign values.

Expressions [page 56]
An expression is a clause that can be evaluated to return values.

Predicates [page 61]

Session Variables [page 75]

SQL Functions [page 83]
Documents the built-in SQL functions that are provided with SAP HANA.

SQL Statements [page 430]
SAP HANA supports many SQL statements to allow you to perform such tasks as create database objects, administer your system, and manipulate data.

System Limitations [page 1208]
Limitations to take into consideration when administering an SAP HANA database.

SQL Error Codes [page 1212]
Each SAP HANA error is identified by a numeric error code and has a short descriptive text.

ANSI SQL Compliance [page 1272]
Information about SAP HANA’s compliance with the mandatory features of the ANSI SQL/2016 standard.

4.1 Introduction to SQL

This chapter describes the SAP HANA database implementation of Structured Query Language (SQL).

• SQL [page 28]
SQL

SQL stands for Structured Query Language. It is a standardized language for communicating with a relational database. SQL is used to retrieve, store or manipulate information in the database.

SQL statements perform the following tasks:

- Schema definition and manipulation
- Data manipulation
- System management
- Session management
- Transaction management

Supported languages and code pages

The SAP HANA database supports Unicode to allow the use of all languages in the Unicode Standard and 7 Bit ASCII code page without restriction.

Comments

You can add comments to improve the readability and maintainability of your SQL statements. Comments are delimited in SQL statements as follows:

- Double hyphens "--". Everything after the double hyphen until the end of a line is ignored by the SQL parser.
- "/*" and "/". This style of commenting is used to place comments on multiple lines. All text between the opening "/*" and closing "/" is ignored by the SQL parser.

Identifiers

Identifiers are used to represent names used in SQL statement including table name, view name, synonym name, column name, index name, function name, procedure name, user name, role name, and so on. There are two kinds of identifiers, undelimited identifiers and delimited identifiers.
• Undelimited table and column names must start with a letter and contain only letters, digits, or underscores "_".
• Delimited identifiers are enclosed in the delimiter, double quotes. The identifier can then contain any character including special characters. "AB$%CD" is a valid identifier name for example.
• Limitations:
  • ", SYS," is reserved exclusively for database engine and is therefore not allowed at the beginning of schema object names.
  • The role name and user name must be specified as undelimited identifiers.
  • The maximum length for identifiers is 127 characters.

Identifiers and case sensitivity

Identifiers without double-quotes in SQL syntax are converted to upper case when processed by the server. For example, the statement `CREATE COLUMN TABLE MyTAB...` creates a table called MYTAB, whereas `CREATE COLUMN TABLE "MyTab"` creates a table called MyTab—and both tables can co-exist in the database.

Specifying identifiers without double-quotes is allowed but can cause ambiguity later when querying or performing operations on objects where casing in the identifier name is significant. A recommendation is to standardize to using double-quotes around all identifiers in SQL statements where ambiguity may be a concern.

User names and passwords

User names may contain underscores and hyphens without requiring quotes. For a list of unpermitted characters in user names, see Unpermitted Characters in User Names topic in the HANA Database Admin Guide.

The following user name formats are unsupported:
• User names starting with double hyphens (for example, --TESTUSER)
• User names containing only number signs (#) and hyphens (for example, #--#--)

Passwords must be a minimum length of 8. For more information on passwords see the Password Policy Configuration Options topic in the HANA Database Admin Guide

Quotation marks

Single quotation marks are used to delimit string literals. A single quotation mark itself can be represented using two single quotation marks. Double quotation marks are used to delimit identifiers. A double quotation mark itself can be represented using two double quotation marks.
4.2 SQL Notation Conventions

SQL syntax notation conventions used in this guide.

This reference uses syntax notation very similar to BNF (Backus-Naur Form), a notation technique used to define programming languages.

Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&gt;</td>
<td>Angle brackets are used to surround the name of a syntactic element (BNF nonterminal) of the SQL language.</td>
</tr>
<tr>
<td>::=</td>
<td>The definition operator is used to provide definitions of the element appearing on the left side of the operator in a production rule.</td>
</tr>
<tr>
<td>[]</td>
<td>Square brackets are used to indicate optional elements in a formula. Optional elements can be specified or omitted.</td>
</tr>
<tr>
<td>{}</td>
<td>Braces group elements in a formula. Repetitive elements (zero or more elements) can be specified within brace symbols.</td>
</tr>
<tr>
<td></td>
<td>The alternative operator indicates that the portion of the formula following the bar is an alternative to the portion preceding it.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>[...]</td>
<td>Ellipsis with square brackets around it indicates optional repetition of the preceding element or grouped elements. For example, if you can specify one or more columns for an option and must separate them by commas, this is expressed as <code>&lt;column_name&gt; [, ...]</code>. An example of grouped elements where a comma separator is not required looks like this: <code>{ &lt;column_name&gt; &lt;data_type&gt; } [...]</code></td>
</tr>
</tbody>
</table>

!! | Introduces normal English text. This is used when the definition of a syntactic element is not expressed in BNF. |

### Lowest Terms Representations

Throughout this manual, each syntax term is defined to one of the lowest term representations shown below.

```
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
<any_character> ::= !!any character.
<comma> ::= ,
<dollar_sign> ::= $
<double_quotes> ::= "
<greater_than_sign> ::= >
<hash_symbol> ::= #
<left_bracket> ::= [
<left_curly_bracket> ::= {
<lower_than_sign> ::= <
<period> ::= .
<pipe_sign> ::= |
<right_bracket> ::= ]
<right_curly_bracket> ::= }
<sign> ::= + | -
<single_quote> ::= '
<underscore> ::= _
<apostrophe> ::= <single_quote>
```
<approximate_numeric_literal> ::= <mantissa>E<exponent>

<cesu8_restricted_characters> ::= <double_quote> | <dollar_sign> | <single_quote> | <period> | <greater_than_sign> | <lower_than_sign> | <pipe_sign> | <left_bracket> | <right_bracket> | <left_curly_bracket> | <right_curly_bracket> | ( | ) | ! | % | * | , | / | : | ; | = | ? | @ | \ | ^ | `.

<exact_numeric_literal> ::= <unsigned_integer>[<period][<unsigned_integer>]]
| <period><unsigned_integer>
<exponent> ::= <signed_integer>

<hostname> ::= {<letter> | <digit>}{[<letter> | <digit> | <period> | - ] [\]} [...]

<identifier> ::= <simple_identifier> | <special_identifier>

<mantissa> ::= <exact_numeric_literal>

<numeric_literal> ::= <signed_numeric_literal> | <signed_integer>

<password> ::= {
             | <letter> [ [ <letter_or_digit> | # | $ ][\] ]
             | <digit> [ [ <letter_or_digit> [\] ]
             | <special_identifier> }

<port_number> ::= <unsigned_integer>

<schema_name> ::= <unicode_name>

<simple_identifier> ::= {<letter> | <underscore>} [{<letter> | <digit> | <underscore> | <hash_symbol> | <dollar_sign>}[...]

<special_identifier> ::= <double_quotes><any_character>...<double_quotes>

<signed_integer> ::= [<sign>]<unsigned_integer>

<signed_numeric_literal> ::= [<sign>]<unsigned_numeric_literal>

<string_literal> ::= <single_quote>[<any_character>...]<single_quote>

<unicode_name> ::= !! CESU-8 string excluding any characters listed in
<cesu8_restricted_characters>

<unsigned_integer> ::= <digit>... 

<unsigned_numeric_literal> ::= <exact_numeric_literal> | 
<approximate_numeric_literal>

<user_name> ::= <unicode_name>
4.3 Data Types

A data type defines the characteristics of a data value. A special value of NULL is included in every data type to indicate the absence of a value.

Classification of Data Types

In the SAP HANA database, each data type can be classified by its characteristics as follows:

<table>
<thead>
<tr>
<th>Classification</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datetime types</td>
<td>DATE, TIME, SECONDDATE, TIMESTAMP</td>
</tr>
<tr>
<td>Numeric types</td>
<td>TINYINT, SMALLINT, INTEGER, BIGINT, SMALLDECIMAL, DECIMAL, REAL, DOUBLE</td>
</tr>
<tr>
<td>Boolean type</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>Character string types</td>
<td>VARCHAR, NVARCHAR, ALPHANUM, SHORTTEXT</td>
</tr>
<tr>
<td>Binary types</td>
<td>VARBINARY</td>
</tr>
<tr>
<td>Large Object types</td>
<td>BLOB, CLOB, NCLOB, TEXT</td>
</tr>
<tr>
<td>Multi-valued types</td>
<td>ARRAY</td>
</tr>
<tr>
<td>Spatial types</td>
<td>ST_GEOMETRY, ST_POINT</td>
</tr>
</tbody>
</table>

Typed Constants

A constant is a symbol that represents a specific fixed data value.

**Character string constant**

A character string constant is enclosed in single quotation marks, for example: 'Brian' or '100'.

Unicode strings have a similar format to character strings but are preceded by an N identifier (N stands for National Language in the SQL-92 standard).

```
SELECT 'Brian' "character string 1", '100' "character string 2", N'abc' "unicode string" FROM DUMMY;
```
The example above returns the following results:

<table>
<thead>
<tr>
<th>character String 1</th>
<th>character String 2</th>
<th>unicode string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian</td>
<td>100</td>
<td>abc</td>
</tr>
</tbody>
</table>

**Number constant**

A number constant is represented by a string of numbers that are not enclosed in quotation marks. Numbers may contain a decimal point or a scientific notation. For example, 123, 123.4, or 1.234e2.

A hexadecimal number constant is a string of hexadecimal numbers and has the prefix 0x. For example, 0x0abc.

```
SELECT 123 "integer", 123.4 "decimal1", 1.234e2 "decimal2", 0x0abc "hexadecimal" FROM DUMMY;
```

The example above returns the following results:

<table>
<thead>
<tr>
<th>integer</th>
<th>decimal1</th>
<th>decimal2</th>
<th>hexadecimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>123.4</td>
<td>123.4</td>
<td>2,748</td>
</tr>
</tbody>
</table>

**Binary string constant**

A binary string has the prefix X and is a string of hexadecimal numbers that are enclosed in quotation marks. For example, X'00abcd' or x'dcba00'.

```
SELECT X'00abcd' "binary string 1", x'dcba00' "binary string 2" FROM DUMMY;
```

The example above returns the following results:

<table>
<thead>
<tr>
<th>binary string 1</th>
<th>binary string 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>00ABCD</td>
<td>DCBA00</td>
</tr>
</tbody>
</table>

**Date/Time/Timestamp constant**

Date, Time, and Timestamp each have the following prefixes:

- date'2010-01-01'
- time'11:00:00.001'
- timestamp'2011-12-31 23:59:59'

```
SELECT date'2010-01-01' "date", time'11:00:00.001' "time",
timestamp'2011-12-31 23:59:59' "timestamp" FROM DUMMY;
```

The example above returns the following results:

<table>
<thead>
<tr>
<th>date</th>
<th>time</th>
<th>timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 1, 2010</td>
<td>11:00:00 AM</td>
<td>Dec 31, 2011 11:59:59.0 PM</td>
</tr>
</tbody>
</table>
4.3.1 Binary Data Types

Binary types are used to store bytes of binary data.

A value of type binary can be converted to a value of type (N)VARCHAR if its size is smaller than or equal to 8192. It can therefore be used like a value of type (N)VARCHAR except for full text search operations and numeric operations.

**VARBINARY**

The VARBINARY(<n>) data type is used to store binary data of a specified maximum length in bytes, where <n> indicates the maximum length and is an integer between 1 and 5000. If the length is not specified, then the default is 1.

4.3.2 Boolean Data Type

The BOOLEAN data type stores boolean values, which are TRUE, FALSE, and UNKNOWN, where UNKNOWN is a synonym of NULL.

When the client does not support a boolean type, it returns 1 for TRUE and 0 for FALSE.

The following example returns TRUE or 1 for boolean:

```sql
CREATE ROW TABLE TEST (A BOOLEAN);
INSERT INTO TEST VALUES (TRUE);
INSERT INTO TEST VALUES (FALSE);
INSERT INTO TEST VALUES (UNKNOWN);
INSERT INTO TEST VALUES (NULL);
SELECT A "boolean" FROM TEST WHERE A = TRUE;
```

Although predicates and boolean expressions can both have the same values (TRUE, FALSE, UNKNOWN), they are not the same. Therefore, you cannot use boolean type comparisons to compare predicates, or use predicates where boolean expressions should be used.

For example, the following statement does not work:

```sql
SELECT * FROM DUMMY WHERE ( 'A'>'B' ) = ( 'C'>'D' );
```

The following statement is the correct way to achieve the results desired from the statement above:

```sql
SELECT * FROM DUMMY WHERE
    CASE WHEN ( 'A'>'B' ) THEN TRUE WHEN NOT ( 'A'>'B' ) THEN FALSE ELSE NULL END =
    CASE WHEN ( 'C'>'D' ) THEN TRUE WHEN NOT ( 'C'>'D' ) THEN FALSE ELSE NULL END;
```

Related Information

TO_BOOLEAN Function (Data Type Conversion) [page 367]
4.3.3 Character String Data Types

Character string data types are used to store values that contain character strings. VARCHAR data types contain 7-bit ASCII character strings, and NVARCHAR are used for storing Unicode character strings.

Collation expressions are not supported, and values of string data type are compared using a binary comparison.

Character string data types in SAP HANA use 7-bit ASCII. Extended ASCII characters are converted into corresponding Unicode characters. If the data includes anything other than 7-bit ASCII characters, use Unicode character string types, such as NVARCHAR and NCLOB.

i Note
The SAP HANA database does not officially support the CHAR and NCHAR datatypes. Even though they are available for use, they are only for legacy support and consistent behavior is not guaranteed for values of these types between a row table and a column table. Use VARCHAR and NVARCHAR instead.

VARCHAR
The VARCHAR(\textless n\textgreater ) data type specifies a variable-length character string, where \textless n\textgreater  indicates the maximum length in bytes and is an integer between 1 and 5000. If the length is not specified in DDL statements, then the default of 1 is used.

If the VARCHAR(\textless n\textgreater ) data type is used in a DML query, for example \texttt{CAST (A as VARCHAR(n))}, then \textless n\textgreater  indicates the maximum length of the string in characters. If the length is not specified, then the default of 5000 is used. For non 7-bit ASCII character-based strings, use NVARCHAR.

NVARCHAR
The NVARCHAR(\textless n\textgreater ) data type specifies a variable-length Unicode character set string, where \textless n\textgreater  indicates the maximum length in characters and is an integer between 1 and 5000. If the length is not specified in DDL statements, then the default of 1 is used.

If the NVARCHAR(\textless n\textgreater ) data type is used in a DML query, for example \texttt{CAST (A as NVARCHAR(n))}, then \textless n\textgreater  indicates the maximum length of the string in characters. If the length is not specified, then the default of 5000 is used. For 7-bit ASCII character-based strings only, use VARCHAR.

ALPHANUM
The ALPHANUM(\textless n\textgreater ) data type specifies a variable-length character string that contains alpha-numeric characters, where \textless n\textgreater  indicates the maximum length and is an integer between 1 and 127.

Sorting among values of type ALPHANUM is performed in alpha-representation. In the case of a purely numeric value, this means that the value can be considered as an alpha value with leading zeros.

SHORTTEXT
The SHORTTEXT(\textless n\textgreater ) data type specifies a variable-length character string that supports text search features and string search features. This data type can be defined for column tables but not for row tables. This is not a standalone SQL type. Selecting a SHORTTEXT(\textless n\textgreater ) column yields a column of type NVARCHAR(\textless n\textgreater ).

\[
\text{SHORTTEXT} \ ::= \text{SHORTTEXT} \{ \text{<unsigned_integer>} \} \quad \text{<elem_list_shorttext>}
\]

\[
\text{<elem_list_shorttext>} \ ::= \text{<fulltext_elem>} \{
\text{<fulltext_elem>} \}...
\]
4.3.4 Data Type Conversion

Both implicit and explicit data type conversions are allowed in the SAP HANA database.

**i Note**

SAP HANA does not support conversions between LOB data types.

**Explicit type conversion**

The data type of an expression result – for example, a field reference, a function on fields, or literals – can be converted using the following functions:

- CAST function
- TO_ALPHANUM function
- TO_BIGINT function
- TO_VARBINARY function
- TO_BLOB function
- TO_CLOB function
- TO_DATE function
- TO_DATS function
- TO_DECIMAL function
- TO_DOUBLE function
- TO_INTEGER function
- TO_INT function
- TO_NCLOB function
- TO_NVARCHAR function
- TO_REAL function
- TO_SECONDDATE function
- TO_SMALLINT function
- TO_TINYINT function
- TO_TIME function
- TO_TIMESTAMP function
- TO_VARCHAR function

**Implicit type conversion**

When a given set of operand/argument types does not match what an operator/function expects, a type conversion is carried out by the SAP HANA database. This conversion only occurs if a relevant conversion is available and if it makes the operation/function executable. For example, a comparison of BIGINT and VARCHAR is performed by implicitly converting VARCHAR to BIGINT. With the exception of TIME and TIMESTAMP data types, explicit conversions can be used for implicit conversions. TIME and TIMESTAMP data types can be converted reciprocally by using the TO_TIME(TIMESTAMP) and TO_TIMESTAMP(TIME) functions.
**Implicit Type Conversion Examples**

<table>
<thead>
<tr>
<th>Input Expression</th>
<th>Transformed Expression with Implicit Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT &gt; VARCHAR</td>
<td>BIGINT &gt; BIGINT(VARCHAR)</td>
</tr>
<tr>
<td>BIGINT &gt; DECIMAL</td>
<td>DECIMAL(BIGINT) &gt; DECIMAL</td>
</tr>
<tr>
<td>TIMESTAMP &gt; DATE</td>
<td>TIMESTAMP &gt; TIMESTAMP(DATE)</td>
</tr>
<tr>
<td>DATE &gt; TIME</td>
<td>Returns an error because conversion cannot occur between DATE and TIME</td>
</tr>
</tbody>
</table>

The rules shown in the tables below are applicable to both implicit and explicit conversion except for TIME to TIMESTAMP conversion. Only explicit conversions are allowed for converting the TIME data type to the TIMESTAMP data type using the TO_TIMESTAMP or CAST functions.

In the tables below, **OK** means that data type conversions are allowed without any checks; **CHK** means that the data type can be converted if the data is valid for the target type; and - means that data type conversion is not allowed.

**Numeric Data Type Conversion table**

<table>
<thead>
<tr>
<th>Target/ Source</th>
<th>INTYINT</th>
<th>SMALL INT</th>
<th>INTEGRER</th>
<th>BIGINT</th>
<th>DECIMAL</th>
<th>DECIMAL(&lt;p&gt;,&lt;s&gt;)</th>
<th>SMALL DECI-MAL</th>
<th>REAL</th>
<th>DOU-BLE</th>
<th>VAR-CHAR</th>
<th>NVARC-HAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI-INYINT</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>SMALL-INT</td>
<td>CHK</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>INTEGRER</td>
<td>CHK</td>
<td>CHK</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>BIGINT</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>-</td>
<td>OK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>DECIMAL(&lt;p&gt;,&lt;s&gt;)</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>SMALL-DECIMAL</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>OK</td>
<td>CHK</td>
<td>-</td>
<td>CHK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>REAL</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>OK</td>
<td>CHK</td>
<td>CHK</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>DOUBBLE</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>VAR-CHAR</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>NVARC-HAR</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>
## Datetime Data Type Conversion Table

<table>
<thead>
<tr>
<th>Target/ Source</th>
<th>TIME</th>
<th>DATE</th>
<th>SECONDDATE</th>
<th>TIMESTAMP</th>
<th>VARCHAR</th>
<th>NVARCHAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>DATE</td>
<td>-</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SECONDDATE</td>
<td>TIME</td>
<td>DATE</td>
<td>-</td>
<td>TIMESTAMP</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIME</td>
<td>DATE</td>
<td>SECONDDATE</td>
<td>-</td>
<td>OK</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>-</td>
<td>OK</td>
</tr>
<tr>
<td>NVARCHAR</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>CHK</td>
<td>OK</td>
<td>-</td>
</tr>
</tbody>
</table>

## Character String Data Type Conversion Table

<table>
<thead>
<tr>
<th>Target/ Source</th>
<th>VARBINARY</th>
<th>VARCHAR</th>
<th>NVARCHAR</th>
<th>ALPHANUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARBINARY</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ALPHANUM</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
<td>-</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>OK</td>
<td>-</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>NVARCHAR</td>
<td>OK</td>
<td>OK</td>
<td>-</td>
<td>OK</td>
</tr>
</tbody>
</table>

## Data Type Precedence

Data type precedence specifies that the data type with the lower precedence is converted to the data type with the higher precedence.

The following list specifies the precedence of data types, with TIMESTAMP being the highest and VARBINARY being the lowest:

- TIMESTAMP
- SECONDDATE
- DATE
- TIME
- DOUBLE
- REAL
- DECIMAL
- SMALLDECIMAL
- DECIMAL(<precision>,<scale>)
- BIGINT
- INTEGER
- SMALLINT
- TINYINT
- NCLOB
- NVARCHAR
- CLOB
- VARCHAR
Spatial Data Type Conversion

Spatial data types can be converted using an explicit data type conversion function (TO_CHAR, TO_NCHAR, TO_VARCHAR, TO_NVARCHAR, TO_CLOB, TO_NCLOB, TO_BINARY, TO_VARBINARY). Implicit data conversion for spatial data types is not supported.

The following examples show how the data type conversion functions must be used to convert spatial data types:

```
Select TO_CHAR(NEW ST_Point('Point M( 1.0 2.0 NULL)')) from dummy;
Select TO_BINARY(NEW ST_Point('Point M( 1.0 2.0 NULL)')) from dummy;
Select TO_VARBINARY(NEW ST_Point('Point M( 1.0 2.0 NULL)')) from dummy;
-- and so on
```

Example

- When converting numeric types to DECIMAL or SMALLDECIMAL types, the least significant digits are rounded. For example, converting a BIGINT value such as 1234567890123456789 to SMALLDECIMAL becomes 1234567890123457000.
- When converting numeric types to any other numeric types, the least significant digits are truncated toward 0. The most significant digits cannot be truncated. Overflow errors occur if the converted value is incompatible with the target format. For example, converting a DECIMAL(10,2) value such as 12345.67 to BIGINT becomes 12345.
- When converting character string types to numeric types, no truncation is allowed, and an overflow error or invalid number error can be returned. For example, converting a VARCHAR value such as "12345.67" to BIGINT fails and returns an invalid number error. And converting a the VARCHAR value "256" to TINYINT generates and overflow error.
- When converting datetime data types to other datetime data types, the results are truncated to the target type when the source type is larger; otherwise, the default value is added. For example, you can converting the TIMESTAMP 2013-07-03 01:23:45.123456789 to a DATE data type returns 2013-07-03. Converting the DATE 2013-07-03 to a TIMESTAMP returns 2013-07-03 00:00:00.000000000. Converting the TIMESTAMP 2013-07-03 23:59:59.7777777 to a SECONDDATE returns 2013-07-03 23:59:59.
- When converting character string types to datetime types, the results are truncated to the target type, and an invalid DATE, TIME of TIMESTAMP error can be returned. For example, converting the VARCHAR value "2013-07-03 00:00:00.000000000" to a DATE returns 2013-07-03.

4.3.5 Datetime Data Types

Datetime data types are used to store date and time information.

**DATE**
The DATE data type consists of year, month, and day information to represent a date value. The default format for the DATE data type is YYYY-MM-DD. YYYY represents the year, MM represents the month, and DD represents the day. The range of the date value is between 0001-01-01 and 9999-12-31.

DATE is a synonym for DAYDATE.

**TIME**

The TIME data type consists of hour, minute, and second information to represent a time value. The default format for the TIME data type is HH24:MI:SS. HH24 represents the hour from 0 to 24, MI represents the minute from 0 to 59, and SS represents the second from 0 to 59.

**SECONDDATE**

The SECONDDATE data type consists of year, month, day, hour, minute, and second information to represent a date with a time value. The default format for the SECONDDATE data type is YYYY-MM-DD HH24:MI:SS. YYYY represents the year, MM represents the month, DD represents the day, HH24 represents hours, MI represents minutes, and SS represents seconds. The range of the date value is between 0001-01-01 00:00:01 and 9999-12-31 24:00:00.

**TIMESTAMP**

The TIMESTAMP data type consists of date and time information. Its default format is YYYY-MM-DD HH24:MI:SS.FF<sup>n</sup>. FF<sup>n</sup> represents the fractional seconds where <sup>n</sup> indicates the number of digits in fractional part. The range of the time stamp value is between 0001-01-01 00:00:00.0000000 and 9999-12-31 23:59:59.9999999. LONGDATE is a synonym of TIMESTAMP.

The EMPTY value refers to the lowest possible value of each datetime type and ensures compatibility with ABAP.

**Datetime Session Variables**

The following datetime session variables can be used to override the system default datetime formats for the session. Use the SET statements to set session variables.

**DATE_FORMAT**

Specifies the default DATE format to apply for the session.

**TIME_FORMAT**

Specifies the default TIME format to apply for the session.

**SECONDDATE_FORMAT**

Specifies the default SECONDDATE format to apply for the session.

**TIMESTAMP_FORMAT**

Specifies the default TIMESTAMP format to apply for the session.

When setting the TIMESTAMP_FORMAT session variables, if you do not want to lose decimal point seconds when converting values, then specify SS.FF7 as the format for seconds instead of SS.

**Example of how setting the session variable affects a default format:**

```sql
CREATE TABLE T1 (A TIMESTAMP);
INSERT INTO T1 VALUES ('2018/01/02 10:00:00');     --> OK
INSERT INTO T1 VALUES ('02/01/2018 10:00:00');     --> ERROR
SELECT TO_VARCHAR(A) FROM T1;                   --> 2018-01-02 10:00:00.0000000
```
SET 'TIMESTAMP_FORMAT' = 'DD/MM/YYYY HH:MI:SS';
CREATE TABLE T2 (A TIMESTAMP);
INSERT INTO T2 VALUES ('2018/01/02 10:00:00'); --> ERROR
INSERT INTO T2 VALUES ('02/01/2018 10:00:00'); --> OK
SELECT TO_VARCHAR(A) FROM T2; --> 02/01/2018 10:00:00

Date Formats

The following datetime formats can be used when parsing a string into a DATETIME type and converting a DATETIME type value into a string value. The format for TIMESTAMP is the combination of DATE and TIME with additional support for fractional seconds. An empty date (0000-00-00) is a special value in SAP HANA. Even though an empty date looks like a NULL or unknown value, it is not. For example, the empty date can be represented as '', which behaves like an empty string. It also satisfies a IS NOT NULL predicate. Use the following statements to test the behavior of the empty date value:

CREATE ROW TABLE T (A INT, B DATE, C DATE);
INSERT INTO T VALUES (1, '', '0001-01-01');
INSERT INTO T VALUES (2, '0000-00-00', '0001-01-01');
INSERT INTO T VALUES (3, '0000-00-00', '0001-01-01');
INSERT INTO T VALUES (4, '0001-01-01', '0001-01-01');
INSERT INTO T VALUES (5, NULL, '0001-01-01');
SELECT * FROM T WHERE B = '00000000';
SELECT * FROM T WHERE B = '';
SELECT * FROM T WHERE B IS NOT NULL;
SELECT * FROM T;
SELECT DAYS_BETWEEN(B, C) FROM T;

Supported Formats for DATE

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYYY-MM-DD</td>
<td>Default format.</td>
<td>INSERT INTO my_tbl VALUES ('1957-06-13');</td>
</tr>
<tr>
<td>YYYY/MM/DD</td>
<td>YYYY is from 0001 to 9999, MM is from 1 to 12, and DD is from 1 to 31. If year has less than four digits, month has less than two digits, or day has less than two digits, then values are padded by one or more zeros. For example, a two-digit year like 45 is saved as year 0045, while a one digit month like 9 is saved as 09, and a one digit day like 2 is saved as 02.</td>
<td>INSERT INTO my_tbl VALUES ('1957-06-13');</td>
</tr>
<tr>
<td>YYYY/MM-DD</td>
<td></td>
<td>INSERT INTO my_tbl VALUES ('1957-06-13');</td>
</tr>
<tr>
<td>YYYY-MM/DD</td>
<td>The ABAP Data Type, DATES format.</td>
<td>INSERT INTO my_tbl VALUES ('19570613');</td>
</tr>
<tr>
<td>Format</td>
<td>Description</td>
<td>Examples</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>MON</td>
<td>The abbreviated name of month. For example, JAN. or DEC.</td>
<td><code>INSERT INTO my_tbl VALUES (TO_DATE('2040-Jan-10', 'YYYY-MON-DD'));</code>&lt;br&gt;<code>INSERT INTO my_tbl VALUES (TO_DATE('Jan-10', 'MON-DD'));</code></td>
</tr>
<tr>
<td>MONTH</td>
<td>The name of month. For example, JANUARY - DECEMBER.</td>
<td><code>INSERT INTO my_tbl VALUES (TO_DATE('2040-January-10', 'YYYY-MONTH-DD'));</code>&lt;br&gt;<code>INSERT INTO my_tbl VALUES (TO_DATE('January-10', 'MONTH-DD'));</code></td>
</tr>
<tr>
<td>RM</td>
<td>The Roman numeral month (I-XII; JAN = I).</td>
<td><code>INSERT INTO my_tbl VALUES (TO_DATE('2040-I-10', 'YYYY-RM-DD'));</code>&lt;br&gt;<code>INSERT INTO my_tbl VALUES (TO_DATE('I-10', 'RM-DD'));</code></td>
</tr>
<tr>
<td>DDD</td>
<td>The day of the year (1-366).</td>
<td><code>INSERT INTO my_tbl VALUES (TO_DATE('204', 'DDD'));</code>&lt;br&gt;<code>INSERT INTO my_tbl VALUES (TO_DATE('2001-204', 'YYY-Y-DDD'));</code></td>
</tr>
</tbody>
</table>

**Time Formats**

Supported formats for TIME

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>HH24:MI:SS</td>
<td>The default format.</td>
<td></td>
</tr>
</tbody>
</table>

PUBLIC 43
### Format

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>HH:MI[:SS][AM</td>
<td>PM]</td>
<td>HH is from 0 to 23. MI is from 0 to 59. SS is from 0 to 59.</td>
</tr>
<tr>
<td>HH12:MI:SS[AM</td>
<td>PM]</td>
<td>If a one digit hour, minute, or second is specified, then 0 is also inserted into the value. For example, 9:9:9 is saved as 09:09:09.</td>
</tr>
<tr>
<td>HH24:MI:SS</td>
<td>HH12 indicates a 12 hour clock. HH24 indicates a 24 hour clock.</td>
<td>INSERT INTO my_tbl VALUES ('9:9:9 AM');</td>
</tr>
<tr>
<td></td>
<td>Specify AM or PM as a suffix to indicate that the time value is before or after midday.</td>
<td>INSERT INTO my_tbl VALUES (TO_TIME('11:59:59', 'HH12:MI:SS'));</td>
</tr>
</tbody>
</table>

| SSSSS          | The seconds past midnight (0-86399).                                        | INSERT INTO my_tbl VALUES (TO_TIME('12345', 'SSSS'));                    |

### Timestamp Formats

**Supported formats for TIMESTAMP**

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYYY-MM-DD HH24:MI:SS.FF7</td>
<td>The default format.</td>
<td>INSERT INTO my_tbl VALUES (TO_TIMESTAMP('2011-05-11 12:59.999', 'YYYY-MM-DD HH:MI:SS.FF'));</td>
</tr>
</tbody>
</table>

**FF [1..7]**

Fractional seconds have the range of 1 to 7 after the FF parameter to specify the number of digits in the fractional second portion of the datetime value returned. If a digit is not specified, then an error is returned.

INSERT INTO my_tbl VALUES (TO_TIMESTAMP('2011-05-11 12:59.999', 'YYYY-MM-DD HH:MI:SS.FF'));

### Additional Formats for DATETIME

The following additional formats for DATETIME are supported.

While all of the formats below can be converted to strings (TO VARCHAR function), conversion from string into D, DAY, DY, Q, W, and WW formats (for example, using TO_DATE and TO_TIMESTAMP functions) is not supported.
<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>The day of the week (1-7).</td>
</tr>
<tr>
<td>DAY</td>
<td>The name of the day (MONDAY - SUNDAY).</td>
</tr>
<tr>
<td>DY</td>
<td>The abbreviated name of the day (MON - SUN).</td>
</tr>
<tr>
<td>MON</td>
<td>The abbreviated month name (JAN - DEC).</td>
</tr>
<tr>
<td>MONTH</td>
<td>The full month name (JANUARY - DECEMBER).</td>
</tr>
<tr>
<td>RM</td>
<td>The Roman numeral month (I - XII; I is for January).</td>
</tr>
<tr>
<td>Q</td>
<td>The quarter of the year (1, 2, 3, 4).</td>
</tr>
<tr>
<td>W</td>
<td>The week of the month (1-6).</td>
</tr>
<tr>
<td>WW</td>
<td>The week of the year (1-54).</td>
</tr>
</tbody>
</table>

**Related Information**

SET [SESSION] Statement (Session Management) [page 1171]
M_SESSION_CONTEXT System View [page 2133]
Session Variables [page 75]

### 4.3.6 Large Object (LOB) Data Types

LOB (large objects) data types, such as CLOB, NCLOB, and BLOB, are used to store a large amount of data, such as text documents and images.

- **BLOB**
  The BLOB data type is used to store large amounts of binary data. BLOB values can be converted to VARBINARY.

- **CLOB**
  The CLOB data type is used to store large amounts of 7-bit ASCII character data. CLOB values can be converted to VARCHAR.

- **NCLOB**
  The NCLOB data type is used to store a large Unicode character object. NCLOB values can be converted to NVARCHAR.

- **TEXT**
  The TEXT data type enables text search features. This data type can be defined for column tables, but not for row tables. This is not a standalone SQL type. Selecting a TEXT column yields a column of type NCLOB.
A value of type TEXT cannot be converted implicitly to a value of type (N)VARCHAR, and string functions (UPPER, LOWER, and so on) cannot be applied directly to a value of type TEXT. Explicit conversion from a value of type TEXT to a value of type (N)VARCHAR is allowed. String functions can therefore be applied to the converted value.

For columns of type TEXT, the LIKE predicate is not supported.

```
<text_type> ::= TEXT <opt_fulltext_elem_list_text>
<opt_fulltext_elem_list_text> ::= <fulltext_elem> [{, <fulltext_elem>}...]
```

**BINTEXT**

The BINTEXT data type is similar to the TEXT data type and supports text search features, but it is possible to insert binary data. This data type can be defined for column tables, but not for row tables. This is not a standalone SQL type. Selecting a BINTEXT column yields a column of type NCLOB.

For values of type BINTEXT, the same restrictions apply as for values of type TEXT.

```
<bintext_type> ::= BINTEXT <opt_fulltext_elem_list_bintext>
<opt_fulltext_elem_list_bintext> ::= <fulltext_elem> [{, <fulltext_elem>}...]
```

The following syntax rules are common to TEXT, BINTEXT, and SHORTTEXT data types:

```
<fulltext_elem> ::= LANGUAGE COLUMN <column_name>  
                   LANGUAGE DETECTION ( <string_literal_list> )  
                   MIME TYPE COLUMN <column_name>  
                   <change_tracking_elem>  
                   FUZZY SEARCH INDEX <on_off>  
                   PHRASE INDEX RATIO <index_ratio>  
                   CONFIGURATION <string_literal>  
                   SEARCH ONLY <on_off>  
                   FAST PREPROCESS <on_off>  
                   TEXT ANALYSIS <on_off>  
                   MIME TYPE <string_literal>  
                   TOKEN SEPARATORS <string_literal>
<change_tracking_elem> ::=  
                         SYNC [HRONOUS]  
                         [FLUSH [QUEUE] <flush_queue_elem>]  
<flush_queue_elem> ::=  
                       EVERY <integer_literal> MINUTES  
                       EVERY <integer_literal> MINUTES OR AFTER  
                       DOCUMENTS
<integer_literal> DOCUMENTS
```

LOB types support the following operations:

- The LENGTH() function for values of type CLOB/NCLOB/BLOB, which returns the LOB length in bytes.
- The SUBSTR() function for values of type CLOB/NCLOB, which returns the substring of a (N)CLOB value.
- The COALESCE() function.
- The LIKE and CONTAINS predicate for values of type CLOB/NCLOB.
- The IS NULL predicate for values of type CLOB/NCLOB/BLOB.

LOB data types have the following restrictions:

- LOB columns cannot appear in ORDER BY or GROUP BY clauses.
- LOB columns cannot appear in FROM clauses as join predicates.
- LOB columns cannot appear in WHERE clauses as predicates other than LIKE (meaning that no comparison is allowed).
- LOB columns cannot appear in SELECT clauses as aggregate function arguments.
- LOB columns cannot appear in SELECT DISTINCT clauses.
• LOB columns cannot be used in set operations such as EXCEPT, UNION ALL is an exception.
• LOB columns cannot be used as primary keys.
• LOB columns cannot be used in CREATE INDEX statements.
• LOB columns cannot be used in statistics update statements.

Related Information

System Limitations [page 1208]

4.3.7 Multi-valued Data Types

Multi-valued types are used to store collections of values sharing the same data type.

For multi-valued data types (nested data types), SAP HANA supports all basic data types as element data types such as INTEGER, FLOAT, DECIMAL, VARCHAR, and so on. However, it does not support complex or unusual types like TEXT, LOB, and spatial types. SAP HANA also does not support multi-values of multi-values.

ARRAY

The ARRAY type is used to store collections of values sharing the same data type where each element is associated with exactly one ordinal position. Arrays can contain NULL values as elements to indicate the absence of a value. Arrays are immutable: adding, removing, or changing elements is not possible. Arrays with the WITHOUT DUPLICATES constraint cannot have the same non-NULL value more than once.

Supported functions and expressions for the ARRAY type

• `<ARRAY> || <ARRAY>` (concatenation) (SQLScript)
• `<ARRAY>[<index>]` (element access) (SQLScript)
• CARDINALITY Function
• MEMBER_AT Function
• [NOT] MEMBER OF Function
• SUBARRAY Function
• TRIM_ARRAY Function
• UNNEST (SQLScript)

Related Information

Array Variables
UNNEST Function
4.3.8 Numeric Data Types

Numeric data types are used to store numeric information.

Each numeric type below has a maximum value and minimum value. A numeric overflow exception is thrown if a value is smaller than the minimum value or greater than the maximum value. In order to comply with IEEE754, NaN and infinity are not supported. -0.0 is stored as +0.0.

Floating-point data types are stored in the system using binary numbers. The fractional part of these numbers is represented using a combination of 1/2, 1/4, 1/8, 1/16, and so on. For this reason, they cannot completely represent rational numbers with fractional digits. For example, 0.1 cannot be represented exactly by combining these binary fractions. In this case, you obtain inaccurate results when using a floating-point data type. This is the correct behavior for these data types. The following example demonstrates the behavior and returns 4.6999999999 for DOUBLE_SUM:

```sql
SELECT TO_DOUBLE(0.1) + TO_DOUBLE(4.6) AS DOUBLE_SUM FROM DUMMY;
```

For a list of numeric functions, see Numeric Functions [page 421].

TINYINT

The TINYINT data type stores an 8-bit unsigned integer. The minimum value is 0 and the maximum value is 255.

SMALLINT

The SMALLINT data type stores a 16-bit signed integer. The minimum value is -32,768 and the maximum value is 32,767.

INTEGER

The INTEGER data type stores a 32-bit signed integer. The minimum value is -2,147,483,648 and the maximum value is 2,147,483,647.

BIGINT

The BIGINT data type stores a 64-bit signed integer. The minimum value is -9,223,372,036,854,775,808 and the maximum value is 9,223,372,036,854,775,807.

DECIMAL[(precision, scale)] or DEC[(p,s)]

DECIMAL(<p>, <s>) is the SQL standard notation for fixed-point decimals. <p> specifies precision, or the number of total digits (the sum of whole digits and fractional digits). <s> denotes scale, or the number of fractional digits. If a column is defined as DECIMAL(5, 4) for example, the numbers 3.14, 3.1415, 3.141592 are stored in the column as 3.1400, 3.1415, 3.1415, retaining the specified precision(5) and scale(4).

Precision can range from 1 to 38. The scale can range from 0 to <p>. If the scale is not specified, then it defaults to 0.

If precision and scale are not specified, then DECIMAL becomes a floating-point decimal number. In this case, precision and scale can vary within the range of 1 to 34 for precision and -6,111 to 6,176 for scale, depending on the stored value.

For example, 0.0000001234 (1234E-10) has precision 4 and scale 10. 1.0000001234 (10000001234E-10) has precision 11 and scale 10. The value 1234000000 (1234E6) has precision 4 and scale 6.

SMALLDECIMAL
The SMALLDECIMAL data type is a floating-point decimal number. The precision and scale can vary within the range 1-16 for precision and -369-368 for scale, depending on the stored value. SMALLDECIMAL is supported on row and column store.

REAL

The REAL data type specifies a single-precision, 32-bit floating-point number.

DOUBLE

The DOUBLE data type specifies a double-precision, 64-bit floating-point number. The minimum value is \(-1.7976931348623157\times10^{308}\) and the maximum value is \(1.7976931348623157\times10^{308}\). The smallest positive DOUBLE value is \(2.2250738585072014\times10^{-308}\) and the largest negative DOUBLE value is \(-2.2250738585072014\times10^{-308}\).

FLOAT(n)

The FLOAT\((<n>)\) data type specifies a 32-bit or 64-bit real number, where \(<n>\) specifies the number of significant bits and can range between 1 and 53.

If you use the FLOAT\((<n>)\) data type, and \(<n>\) is smaller than 25, then the 32-bit REAL data type is used instead. If \(<n>\) is greater than or equal to 25, or if \(<n>\) is not declared, then the 64-bit DOUBLE data type is used.

4.3.9 Spatial Data Types

Spatial data types are used to store values that contain spatial data, such as points, lines, or polygons.

The following column types are supported in column tables only: ST_Point, ST_Geometry.

The column type ST_Point supports only the two-dimensional spatial data type ST_Point.

The column type ST_Geometry supports the following spatial data types:

- ST_CircularString
- ST_GeometryCollection
- ST_LineString
- ST_MultiLineString
- ST_MultiPoint
- ST_MultiPolygon
- ST_Point
- ST_Polygon

For more information on spatial data types, refer to the SAP HANA Spatial Reference.
4.4 Reserved Words

Reserved words are words which have a special meaning to the SQL parser in the SAP HANA database and cannot be used as when defining an identifier.

The following list provides you with the current reserved words for the SAP HANA database. You can obtain this list by querying the RESERVED_KEYWORDS system view:

```
SELECT * FROM RESERVED_KEYWORDS ORDER BY RESERVED_KEYWORD;
```

In addition to the keywords listed below, avoid using the reserved keywords from the most recent ANSI SQL standard to ensure the compatibility of your SQL statements with subsequent SAP HANA database developments. However, if you do use any of them, we recommend placing them in uppercase and delimiting them with double quotation marks to ensure compatibility.

```
ALL
ALTER
AS
BEFORE
BEGIN
BOTH
CASE
CHAR
CONDITION
CONNECT
CROSS
CUBE
CURRENT_CONNECTION
CURRENT_DATE
CURRENT_SCHEMA
CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_TRANSACTION_ISOLATION_LEVEL
CURRENT_USER
CURRENT_UCTDATE
CURRENT_UCTTIME
CURRENT_UCTTIMESTAMP
CURRVAL
CURSOR
DECLARE
DISTINCT
ELSE
ELSEIF
END
EXCEPT
EXCEPTION
EXEC
FALSE
FOR
FROM
FULL
GROUP
HAVING
IF
IN
INNER
INOUT
INTERSECT
INTO
IS
JOIN
LATERAL
```
4.5 Operators

Use operators to perform arithmetic operations in expressions. Operators can be used for calculation, value comparison, or to assign values.

- Unary and Binary Operators [page 52]
- Operator Precedence [page 52]
- Arithmetic Operators [page 53]
- String Operators [page 53]
- Comparison Operators [page 54]
- Logical Operators [page 54]
- Set Operators [page 55]
### Unary and Binary Operators

Unary and binary operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
</table>
| Unary    | A unary operator applies to one operand or a single value expression. | operator operand | unary plus operator(+)  
|          |           |        | unary negation operator(-)  
|          |           |        | logical negation(NOT)  |
| Binary   | Binary A binary operator applies to two operands or two value expressions. | operand1 operator operand2 | multiplicative operators (*, /)  
|          |           |        | additive operators (+, -)  
|          |           |        | comparison operators (=, !=, <, <=, >, >=)  
|          |           |        | logical operators (AND, OR) |

### Operator Precedence

An expression can use multiple operators. If the number of operators is greater than one, the SAP HANA database evaluates them in order of operator precedence. You can use parentheses to change the order of evaluation, as expressions contained within parentheses are always evaluated first.

If parentheses are not used, the precedence of the various operators is as indicated by the table below. The SAP HANA database evaluates operators with equal precedence from left to right within an expression.

#### SQL operator precedence

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operator</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest</td>
<td>()</td>
<td>parentheses</td>
</tr>
<tr>
<td></td>
<td>+, -</td>
<td>unary positive and negative operation</td>
</tr>
<tr>
<td></td>
<td>*, /</td>
<td>multiplication, division</td>
</tr>
<tr>
<td></td>
<td>+, -</td>
<td>addition, subtraction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>=, !=, &lt;, &lt;=, &gt;, &gt;=, IS NULL, LIKE, BETWEEN</td>
<td>comparison</td>
</tr>
</tbody>
</table>
### Precedence

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT</td>
<td>logical negation</td>
</tr>
<tr>
<td>AND</td>
<td>conjunction</td>
</tr>
<tr>
<td>Lowest</td>
<td>OR</td>
</tr>
</tbody>
</table>

### Arithmetic Operators

You use arithmetic operators to perform mathematical operations, such as adding, subtracting, multiplying, dividing and negation of numeric values.

#### Arithmetic operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;expression&gt;</code></td>
<td>Negation. If the expression is NULL, the result is NULL.</td>
</tr>
<tr>
<td><code>&lt;expression&gt;</code> + <code>&lt;expression&gt;</code></td>
<td>Addition. If either expression is NULL, the result is NULL.</td>
</tr>
<tr>
<td><code>&lt;expression&gt;</code> - <code>&lt;expression&gt;</code></td>
<td>Subtraction. If either expression is NULL, the result is NULL.</td>
</tr>
<tr>
<td><code>&lt;expression&gt;</code> * <code>&lt;expression&gt;</code></td>
<td>Multiplication. If either expression is NULL, the result is NULL.</td>
</tr>
<tr>
<td><code>&lt;expression&gt;</code> / <code>&lt;expression&gt;</code></td>
<td>Division. If either expression is NULL, or if the second expression is 0, an error is returned.</td>
</tr>
</tbody>
</table>

When the result precision is larger than 38 (the largest precision that the DECIMAL(p, s) type can hold), the result type of arithmetic operators is DECIMAL (floating-point decimal number) type.

**The Result Precision and Scale of Arithmetic Operators on DECIMAL(p, s) Types Where e1 is DECIMAL(p1, s2) and e2 is DECIMAL(p2, s2)**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Result Precision</th>
<th>Result Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1 + e2</td>
<td>max(s1, s2) + max(p1-s1, p2-s2) + 1</td>
<td>max(s1, s2)</td>
</tr>
<tr>
<td>e1 - e2</td>
<td>max(s1, s2) + max(p1-s1, p2-s2) + 1</td>
<td>max(s1, s2)</td>
</tr>
<tr>
<td>e1 * e2</td>
<td>p1 + p2</td>
<td>s1 + s2</td>
</tr>
<tr>
<td>e1 / e2</td>
<td>p1· s1 + s2 + max(6, s1 + p2 + 1)</td>
<td>max(6, s1 + p2 + 1)</td>
</tr>
</tbody>
</table>

### String Operators

A concatenation operator combines two items, such as strings, expressions, or constants, into one.
### Concatenation operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>`&lt;expression&gt;</td>
<td></td>
<td>&lt;expression&gt;`</td>
</tr>
</tbody>
</table>

For VARCHAR or NVARCHAR type strings, leading or trailing spaces are retained. If either string is of data type NVARCHAR, the result has data type NVARCHAR and is limited to 64 MB in length. The maximum length for VARCHAR concatenation is also limited to 5000 characters.

### Comparison Operators

**Syntax:**

```sql
<comparison_operation> ::= <expression1> <comparison_operator> <expression2>
```

**Comparison operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>=</code></td>
<td>Equal to</td>
<td>SELECT * FROM students WHERE id = 25;</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Greater than</td>
<td>SELECT * FROM students WHERE id &gt; 25;</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Less than</td>
<td>SELECT * FROM students WHERE id &lt; 25;</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Greater than or equal to</td>
<td>SELECT * FROM students WHERE id &gt;= 25;</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Less than or equal to</td>
<td>SELECT * FROM students WHERE id &lt;= 25;</td>
</tr>
<tr>
<td><code>!=, &lt;&gt;</code></td>
<td>Not equal</td>
<td>SELECT * FROM students WHERE id != 25;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SELECT * FROM students WHERE id &lt;&gt; 25;</td>
</tr>
</tbody>
</table>

### Logical Operators

Search conditions can be combined using AND or OR operators. You can also negate them using the NOT operator.
Logical operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>WHERE condition1 AND condition2</td>
<td>When using AND, the combined condition is TRUE if both conditions are TRUE, FALSE if either condition is FALSE, and UNKNOWN otherwise.</td>
</tr>
<tr>
<td>OR</td>
<td>WHERE condition1 OR condition2</td>
<td>When using OR, the combined condition is TRUE if either condition is TRUE, FALSE if both conditions are FALSE, and UNKNOWN otherwise.</td>
</tr>
<tr>
<td>NOT</td>
<td>WHERE NOT condition</td>
<td>The NOT operator is placed before a condition to negate the condition. The NOT condition is TRUE if condition is FALSE, FALSE if condition is TRUE, and UNKNOWN if condition is UNKNOWN.</td>
</tr>
</tbody>
</table>

Set Operators

Set operators allow multiple queries to be combined to return a single result set.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNION</td>
<td>Combines the results of two or more select statements or query expressions</td>
</tr>
<tr>
<td>UNION ALL</td>
<td>Combines the results of two or more select statements or query expressions, including all duplicate rows.</td>
</tr>
<tr>
<td>INTERSECT</td>
<td>Combines the results of two or more select statements or query expressions, and returns all common rows.</td>
</tr>
<tr>
<td>EXCEPT</td>
<td>Takes the output from the first query and then removes the rows selected by the second query. MINUS is an accepted synonym for EXCEPT.</td>
</tr>
</tbody>
</table>

Related Information

SELECT Statement (Data Manipulation) [page 1104]
4.6 Expressions

An expression is a clause that can be evaluated to return values.

- Case Expressions [page 56]
- Function Expressions [page 57]
- Aggregate Expressions [page 57]
- JSON Object Expressions [page 60]
- Subqueries in Expressions [page 60]

Syntax

```plaintext
<expression> ::= <case_expression> | <function_expression> | <aggregate_expression> ... | <variable_name> | <constant> | [<correlation_name>.]<column_name>
```

Case Expressions

A case expression allows the user to use IF - THEN - ELSE logic without using SQLScript in SQL statements.

Syntax

```plaintext
<case_expression> ::= <simple_case_expression> | <search_case_expression>

<simple_case_expression> ::= CASE <expression>
                         WHEN <expression> THEN <expression>
                         [( WHEN <expression> THEN <expression> )...]
                         [ ELSE <expression> ]
                         END

<search_case_expression> ::= CASE
                          WHEN <condition> THEN <expression>
                          [( WHEN <condition> THEN <expression> )...]
                          [ ELSE <expression> ]
                          END

<condition> ::= <condition> OR <condition> | <condition> AND <condition>
              | NOT <condition> | ( <condition> ) | <predicate>
```

If the expression following the CASE statement is equal to the expression following the WHEN statement, then the expression following the THEN statement is returned. Otherwise, the expression following the ELSE statement is returned if it exists.

While some short-circuiting behavior to improve performance may take place, when and whether all WHEN clauses are evaluated in a CASE expression is not guaranteed.
If all `<expression>` (this includes predicate expressions and/or scalar expressions) do not return an error, then the result of CASE expression is deterministic. That is, the CASE expression will always return a value (returning behavior) and will always return a value from the first matching part (return value behavior).

If any `<expression>` returns an error, then the result of CASE is not deterministic; it may return a value or throw an error depending on an evaluation order of expressions determined by the HANA Optimizer and execution engine. If in this scenario CASE does return a value, then the value will be from the first matching part.

Consider the following example:

```sql
CASE WHEN p1 THEN e1 WHEN p2 THEN e2 ELSE e3 END
```

In SAP HANA, p1, e1, p2, e2 and e3 can be evaluated in any order. But let’s assume for example’s sake that p1 would always return TRUE, and e2 would always throw an exception.

If the order chosen is p1 > e1 > p2 > e2 > e3, and since p1 returns TRUE, then e1 is evaluated and returned as the result of the CASE expression. The remainder of the expressions p2, e2, and e3, are never evaluated so an exception is not thrown for e2.

If the order chosen is e1 > e2 > e3 > p1 > p2, and since e2 returns an exception, then the rest of e3, p1, p2 is never evaluated and an exception is thrown.

Either of these orders and outcomes are possible in SAP HANA, and this is true for both `<simple_case_expression>` and `<search_case_expression>`.

### Function Expressions

SQL built-in functions can be used as expressions.

**Syntax**

```
<function_expression> ::= <function_name> ( <expression> [{, <expression>}...] )
```

### Aggregate Expressions

An aggregate expression uses an aggregate function to calculate a single value from the values of multiple rows in one or more columns.

**Syntax**

```
<aggregate_expression> ::= COUNT(*)
| COUNT ( DISTINCT <expression_list> )
| <agg_name> ( [ ALL | DISTINCT ] <expression> )
| STRING_AGG ( <expression> [, <delimiter>] ) [ <aggregate_order_by_clause> ]
| <agg_name> ::= CORR | CORR_SPEARMAN | COUNT | MIN | MEDIAN | MAX | SUM | AVG | STDDEV | VAR | STDDEV_POP | VAR_POP | STDDEV_SAMP | VAR_SAMP
| <aggregate_order_by_clause> ::= ORDER BY <expression> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST]
```
You can specify to sort the aggregate using the `<aggregate_order_by_clause>`. ASC sorts records in ascending order. DESC sorts records in descending order. By default, for ascending ordering, NULL values are returned first, and for descending they are returned last. You can override this behavior by using NULLS FIRST or NULLS LAST to explicitly specify NULL value ordering.

**Aggregate expressions with DISTINCT:** When you specify the DISTINCT keyword in a query, duplicate values are eliminated before calculations take place. The following aggregate functions are not supported for use with DISTINCT: STDDEV_POP, STDDEV_SAMP, VAR_POP, VAR_SAMP.

<table>
<thead>
<tr>
<th>Aggregate name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORR</td>
<td>Computes the Pearson product momentum correlation coefficient between two columns. See the CORR function for more details.</td>
</tr>
<tr>
<td>CORR_SPEARMAN</td>
<td>Returns the Spearman’s rank correlation coefficient of the values found in the corresponding rows of two columns. See the CORR_SPEARMAN function for more details.</td>
</tr>
<tr>
<td>COUNT</td>
<td>Counts the number of rows returned by a query. COUNT(*) returns the number of rows, regardless of the value of those rows and including duplicate values. COUNT(&lt;expression&gt;) returns the number of non-NULL values for that expression returned by the query. COUNT(DISTINCT &lt;expression_list&gt;) returns the number of distinct values for that expressions returned by the query, excluding rows with all NULL values for that expression.</td>
</tr>
<tr>
<td>MIN</td>
<td>Returns the minimum value of the expression.</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>Finds the statistical median of an input column with a numeric data type. See the MEDIAN function for more information.</td>
</tr>
<tr>
<td>MAX</td>
<td>Returns the maximum value of the expression.</td>
</tr>
<tr>
<td>SUM</td>
<td>Returns the sum of the expression.</td>
</tr>
<tr>
<td>AVG</td>
<td>Returns the arithmetical mean of the expression.</td>
</tr>
<tr>
<td>STDDEV</td>
<td>Returns the standard deviation of the given expression as the square root of the VAR function.</td>
</tr>
<tr>
<td>STDDEV_POP</td>
<td>Returns the standard deviation of the given expression as the square root of the VAR_POP function.</td>
</tr>
<tr>
<td>STDDEV_SAMP</td>
<td>Returns the standard deviation of the given expression as the square root of the VAR_SAMP function.</td>
</tr>
<tr>
<td>Aggregate name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>VAR</td>
<td>Returns the variance of the given expression as the square of the standard deviation.</td>
</tr>
<tr>
<td>VAR_POP</td>
<td>Returns the population variance of expression as the sum of squares of the difference of <code>&lt;expression&gt;</code> from the mean of <code>&lt;expression&gt;</code>, divided by the number of rows remaining.</td>
</tr>
<tr>
<td>VAR_SAMP</td>
<td>Returns the sample variance of expression as the sum of squares of the difference of <code>&lt;expression&gt;</code> from the mean of <code>&lt;expression&gt;</code>, divided by the number of rows remaining minus 1 (one). This function is similar to VAR, the only difference is that it returns NULL when the number of rows is 1.</td>
</tr>
<tr>
<td>STRING_AGG</td>
<td>Returns the concatenated string.</td>
</tr>
</tbody>
</table>

**Result type of numeric aggregate expressions**

<table>
<thead>
<tr>
<th>Aggregate name</th>
<th>tinyint</th>
<th>smallint</th>
<th>integer</th>
<th>bigint</th>
<th>decimal (p,s)</th>
<th>decimal</th>
<th>real</th>
<th>double</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>bigint</td>
<td>bigint</td>
<td>bigint</td>
<td>bigint</td>
<td>bigint</td>
<td>bigint</td>
<td>bigint</td>
<td>bigint</td>
</tr>
<tr>
<td>MIN</td>
<td>tinyint</td>
<td>smallint</td>
<td>integer</td>
<td>bigint</td>
<td>decimal (decimal p,s)</td>
<td>decimal</td>
<td>real</td>
<td>double</td>
</tr>
<tr>
<td>MAX</td>
<td>tinyint</td>
<td>smallint</td>
<td>integer</td>
<td>bigint</td>
<td>decimal (decimal p,s)</td>
<td>decimal</td>
<td>real</td>
<td>double</td>
</tr>
<tr>
<td>SUM</td>
<td>integer</td>
<td>integer</td>
<td>integer</td>
<td>bigint</td>
<td>decimal (decimal p',s)*</td>
<td>decimal</td>
<td>real</td>
<td>double</td>
</tr>
<tr>
<td>AVG</td>
<td>decimal (9,6)</td>
<td>decimal (11,6)</td>
<td>decimal (16,6)</td>
<td>decimal (25,6)</td>
<td>decimal (decimal p,s)</td>
<td>decimal</td>
<td>real</td>
<td>double</td>
</tr>
<tr>
<td>STDDEV</td>
<td>decimal (9,6)</td>
<td>decimal (11,6)</td>
<td>decimal (16,6)</td>
<td>decimal (25,6)</td>
<td>decimal (decimal p,s)</td>
<td>decimal</td>
<td>real</td>
<td>double</td>
</tr>
<tr>
<td>VAR</td>
<td>decimal (9,6)</td>
<td>decimal (11,6)</td>
<td>decimal (16,6)</td>
<td>decimal (25,6)</td>
<td>decimal (decimal p,s)</td>
<td>decimal</td>
<td>real</td>
<td>double</td>
</tr>
</tbody>
</table>

* p' is determined by the following rule: p' = 18 when p <= 18, p' = 28 when p <= 28 and p' = 38 when p <= 38

The following statements return the number of distinct combinations of column A and column B values, excluding any all-NULL tuples.

```sql
CREATE ROW TABLE T (A INT, B INT);
INSERT INTO T VALUES (NULL, NULL);
INSERT INTO T VALUES (1, NULL);
INSERT INTO T VALUES (1, NULL);
INSERT INTO T VALUES (NULL, 1);
INSERT INTO T VALUES (1, 1);
SELECT COUNT (DISTINCT A, B) AS DISTINCT_A_B FROM T;
distinct_a_b
```
Based on the values in the table, there are three distinct combinations (shown in bold below) that are not an all-NULL tuple:

- A=1, B=NULL
- A=NULL, B=1
- A=1, B=1

JSON Object Expressions

The JSON object expression generates a JSON object, and looks very similar to a JSON document, including being wrapped in curly braces. A JSON object comprises one or more key-value pairs, where:

- `<key>` is similar to a string literal and must always be enclosed in double quotes.
- `<value>` is the value for the key and can be any kind of expression, including nested JSON objects enclosed in curly braces (`{}`).

JSON object expressions are only allowed when working with JSON collection tables. They can be referenced by using statements such as SELECT, UPDATE, INSERT INTO, and can be used with operators, such as: +, -, /, *.

Syntax

```
<json_object_expression> ::= {"<key>":<json_value_expression>}
<json_value_expression> ::= '"<string>"'  
<numeric_literal>  
<boolean_literal>  
NULL  
<path_expression>  
<json_object_expression>
<json_array_expression> ::= [ <json_array_value_expression>,... ]
<json_array_value_expression> ::= '"<string>"'
<numeric_literal>  
<boolean_literal>  
NULL  
<path_expression>  
<json_object_expression>
```

The following examples are valid JSON objects:

- `{ "firstname":"John", "lastname":"Smith", "age":45 }`
- `{ "firstname":"John", "lastname":"Smith", "age":45, "address": { 'street': 'Dietmar-Hopp-Allee', 'city': 'Heidelberg' } }`

Subqueries in Expressions

A subquery is a SELECT statement enclosed in parentheses. The SELECT statement can contain no more than one select list item. When used as an expression, a scalar subquery can only return a zero or a single value.
Syntax

\[ \texttt{<scalar_subquery_expression>} ::= (\texttt{<subquery>}) \]

In the SELECT list of the top level SELECT, or in the SET clause of an UPDATE statement, you can use a scalar subquery anywhere where you can use a column name. scalar_subquery cannot be used inside the GROUP BY clause however.

The following statement returns the number of employees in each department for example, grouped by department name:

```
SELECT DepartmentName, COUNT(*), 'out of',
(SELECT COUNT(*) FROM Employees)
FROM Departments AS D, Employees AS E
WHERE D.DepartmentID = E.DepartmentID
GROUP BY DepartmentName;
```

Related Information

Predicates [page 61]
MEDIAN Function (Aggregate) [page 260]
CORR_SPEARMAN Function (Aggregate) [page 148]
CORR Function (Aggregate) [page 145]

4.7 Predicates

A predicate is specified by combining one or more expressions, or logical operators, and returns one of the following logical/truth values: TRUE, FALSE, or UNKNOWN.

Comparison Predicates (ANY, SOME, and ALL) [page 62]
Comparing values using the specified comparison operator and returns true, false, or unknown.

BETWEEN Predicate [page 63]
Comparing a value with a list of values within the specified range and returns true or false.

CONTAINS Predicate [page 64]
Matches a search string with the results of a subquery.

EXISTS Predicate [page 68]
Tests for the presence of a value in a set and returns either true or false.

IN Predicate [page 69]
Searches for a value in a set of values and returns true or false.

LIKE Predicate [page 71]
Performs a comparison to see if a character string matches a specified pattern.

LIKE_REGEXPR Predicate [page 72]
Performs regular expression matching.
MEMBER OF Predicate [page 73]
Determines whether a value is a member of an array.

NULL Predicate [page 74]
Performs a comparison of the value of an expression with NULL.

4.7.1 Comparison Predicates (ANY, SOME, and ALL)

Compares values using the specified comparison operator and returns true, false, or unknown.

Syntax

```
<comparison_predicate> ::= <expression> { = | != | <> | > | < | >= | <= } [ ANY | SOME | ALL ] ( { <expression_list> | <subquery> } )
```

Syntax Elements

**expression_list**

```
<expression_list> ::= <expression> [ ( { <expression> } )... ]
```

An `<expression>` is either a simple expression such as a character, a date or a number, or it can be a scalar subquery.

**ANY | SOME**

Specifies that the comparison returns true if the comparison of the `<expression>` and at least one value returned by the `<subquery>` or `<expression_list>` is true. For example:

```
SELECT * FROM DeptTable WHERE DeptTable.LocColumn = SOME ('BOSTON','DALLAS');
```

If the `<subquery>` or `<expression_list>` is empty, the comparison returns false.

**ANY and SOME, with the equal operator, is the equivalent of IN.**

**ALL**

Specifies that the comparison returns true if the comparison of all values returned by the `<subquery>` or `<expression_list>` is true. If the `<subquery>` or `<expression_list>` is empty, the comparison returns true. For example:

```
SELECT * FROM EmployeeTable WHERE EmployeeTable.Salary >= ALL (1400, 3000);```
Description

Two values are compared using comparison predicates, and the comparison returns true, false, or unknown.

Related Information

SAP HANA SQL Reference Guide (New and Changed)
Expressions [page 56]
SELECT Statement (Data Manipulation) [page 1104]

4.7.2 BETWEEN Predicate

Compares a value with a list of values within the specified range and returns true or false.

Syntax

\[
\text{<between_predicate>} ::= \text{<expression>} \ [\text{NOT}] \ \text{BETWEEN} \ \text{<lower_expression>} \ \text{AND} \ \text{<upper_expression>}
\]

Syntax Elements

expression
The value to search for in the specified list of values.
lower_expression
An expression setting the lower bound of the value list to compare \text{<expression>} to.
upper_expression
An expression setting the upper bound of the value list to compare \text{<expression>} to.

NOT
Inverts the operation of the BETWEEN predicate: returns TRUE when \text{<expression>} is not in the range of values specified between \text{<lower_expression>} and \text{<upper_expression>}, including equal to \text{<lower_expression>} and \text{<upper_expression>}. 
Description

The range predicate returns true if \(<expression1>\) is within the range specified by \(<lower_expression>\) and \(<upper_expression>\), and NOT is not specified.

\[\text{Note}\]

TRUE will only be returned if \(<lower_expression>\) has a value less than or equal to \(<upper_expression>\).

An expression is either a simple expression such as a character, a date or a number, or it can be a scalar subquery.

Related Information

SAP HANA SQL Reference Guide (New and Changed)
Expressions [page 56]
SELECT Statement (Data Manipulation) [page 1104]

4.7.3 CONTAINS Predicate

Matches a search string with the results of a subquery.

Syntax

\[
\text{<contains_predicate>} ::= \text{CONTAINS} \ ( \\
\text{<contains_columns>}, \ \\
\text{<search_string>} \ \\
[], \ <\text{search_specifier}> \ )
\]

Syntax Elements

\text{contains_columns}

Specifies the columns to search.

\[
\text{<contains_columns>} ::= \ * \\
\ | \ ( \ <\text{column_list}> \ ) \\
\text{<column_list>} ::= \ ( \ <\text{column_name}> [,<>] \ )
\]
**search_string**

Specifies the string to search for in `<contains_columns>`, using the freestyle search string format (for example, Peter "Palo Alto" OR Berlin -"SAP LABS").

```plaintext
<search_string> ::= <string_const>
```

**search_specifier**

Defines specifications on the type of matching to perform. If `<search_specifier>` is not specified, EXACT is the default.

```plaintext
<search_specifier> ::= [<search_type>] [<opt_search_specifier2_list>] | <search_specifier2_list>
<opt_search_specifier2_list> ::= (empty, nothing specified) | <search_specifier2_list>
<search_type> ::= <exact_search> | <fuzzy_search> | <linguistic_search>
<search_specifier2> ::= <weights> | <language> | <fulltext>
<fulltext> ::= <FULLTEXT> <(ON)> | <(OFF)> | <(AUTOMATIC)>
<exact_search> ::= EXACT | EXACT ( <additional_params> )
<fuzzy_search> ::= FUZZY | FUZZY ( <fuzzy_params> ) | FUZZY ( <fuzzy_params_list> )
<fuzzy_params_list> ::= ( <fuzzy_params> ), <fuzzy_params_list2>
<fuzzy_params_list2> ::= ( <fuzzy_params> ) | <fuzzy_params_list2>, ( <fuzzy_params> )
<fuzzy_params> ::= <float_const> | <float_const>, <additional_params> | NULL, <additional_params>
<linguistic_search> ::= LINGUISTIC | LINGUISTIC ( <additional_params> )
<weights> ::= WEIGHT ( <float_const_list> )
<language> ::= LANGUAGE ( <string_const> )
<additional_params> ::= <string_const>
FUZZINESS_THRESHOLD
```
If the FUZZINESSTHRESHOLD parameter is defined in a join view, and `<search_specifier>` is not specified, a fuzzy search is performed using the fuzziness threshold defined in the view.

**EXACT**

Specifies to match the term or phrase exactly.

EXACT returns records where exact matches of the search terms are found in the search attributes.

**FUZZY**

Specifies to return matches that are similar to `<search_string>` (spelling errors are ignored to a certain extent for example).

Optionally, you can control the degree of fuzziness using parameters. For example, `<float_const>` specifies the degree of fuzziness expressed as value between 0.0 and 1.0, where 0.0 is very fuzzy, and 1.0 is exact. If `<float_const>` is not specified, 0.8 is the default. It is possible to override this default by defining parameter FUZZINESSTHRESHOLD supported by columnstore join views.

When FUZZY is specified with more than one `<additional_params>` parameter, the number of `<float_const>` / `<additional_params>` pairs must match the number of column names in the column list.

**additional_params**

Specifies additional search parameters in the format of key-value pairs.

**WEIGHT**

Specifies weightings for the columns. If a weights list is specified, it must be the same size as the number of (expanded) columns in `<contains_columns>`.

**LANGUAGE**

Specifies the language characteristics to apply when searching. LANGUAGE is used during preprocessing of the search string and as a pre-search filter. Only documents which match the search string and the language specified are returned.

**LINGUISTIC**

Specifies to perform a linguistic search. A linguistic search finds all words that have the same word stem as the search term. It also finds all words for which the search term is the word stem. For example, searching for ‘cats’ also returns records that contain ‘cat’. You can only perform linguistic searches on columns that contain text, and for which the FAST_PREPROCESS parameter is specified as OFF.

**FULLTEXT ( ON | OFF | AUTOMATIC )**

Specifies the behavior of the CONTAINS predicate in the presence/absence of a full text index:

**ON**

For columns with a full text index a full text search is done. If there is no full text index, the search fails with an error.

**OFF**

When OFF is specified, the CONTAINS predicate behaves as though there is no full text index, even if there is one. That is, if there is a full text index, then it is not used.

**AUTOMATIC**

When AUTOMATIC is specified, the CONTAINS predicate behaves according to whether there is a full text index. That is, if there is a full text index, then it is used. This is the default behavior.
Remarks

The CONTAINS predicate only works on column store objects (column tables and attribute views).

If there are multiple CONTAINS predicates specified in the WHERE clause of a SELECT statement, only one of the predicates can consist of more than one column in the `<contains_columns>` list.

There are many more search parameters (as suggested previously by `<additional_params>`) and examples for full-text search using the CONTAINS predicate found in the SAP HANA Search Developer Guide provided with SAP HANA Advanced Data Processing.

Examples

The following example performs a fuzzy search for the term `cap`, and returns the rows that match (Blue baseball cap, and Red car), in descending order by score:

```sql
CREATE SCHEMA mySchema;
CREATE COLUMN TABLE mySchema.SEARCH_TEXT( Content TEXT FAST PREPROCESS OFF, Descrip TEXT FAST PREPROCESS OFF, Comment TEXT FAST PREPROCESS OFF );
INSERT INTO mySchema.SEARCH_TEXT VALUES( 'Blue baseball cap', 'Vintage', 'Out of stock');
INSERT INTO mySchema.SEARCH_TEXT VALUES( 'Red car', 'Vintage', 'Taking orders' );
INSERT INTO mySchema.SEARCH_TEXT VALUES( 'Bluish sky', 'Retro', 'Discontinued' );
SELECT SCORE() AS SCORE, *
FROM mySchema.SEARCH_TEXT
WHERE CONTAINS(CONTENT,'cap',FUZZY(0.0))
ORDER BY SCORE DESC;
```

Changing SELECT statement to set the FUZZY parameter to 0.9, the equivalent of making the query less fuzzy or more exact, returns only the Blue baseball cap row:

```sql
SELECT SCORE() AS SCORE,*
FROM mySchema.SEARCH_TEXT
WHERE CONTAINS(CONTENT,'cap',FUZZY(0.9))
ORDER BY SCORE DESC;
```

Using the table created in the previous example, the following statement performs an exact term search for either `cap` or `sky`, and returns the Blue baseball cap and Bluish sky rows:

```sql
SELECT * FROM mySchema.SEARCH_TEXT WHERE CONTAINS(CONTENT,'cap OR sky');
```

The following statement performs an exact phrase search for either `baseball cap` and returns the Blue baseball cap row:

```sql
SELECT * FROM mySchema.SEARCH_TEXT WHERE CONTAINS(CONTENT, '"baseball cap"');
```

The following statement shows a LINGUISTIC search. Searching for `take` returns the row that contains Taking orders because `take` is the stem word for `taking` in English.

```sql
SELECT * FROM MySchema.SEARCH_TEXT WHERE CONTAINS (*, 'take', LINGUISTIC);
```
The following statements demonstrate ways to perform a freestyle search over multiple columns. The second example is particularly helpful when you want to search across an entire table without listing the rows.

```sql
SELECT * FROM mySchema.SEARCH_TEXT WHERE CONTAINS ((Content, Descrip), 'vintage');
SELECT * FROM mySchema.SEARCH_TEXT WHERE CONTAINS (*, 'vintage');
```

### Related Information

- Full-Text Search with SQL

### 4.7.4 EXISTS Predicate

Tests for the presence of a value in a set and returns either true or false.

#### Syntax

```sql
<exists.predicate> ::= [NOT] EXISTS ( <subquery> )
```

#### Syntax Elements

- **NOT**
  
  Inverts the operation of the EXISTS predicate: true is returned when the `<subquery>` returns an empty result set and false is returned when the `<subquery>` returns a result set.

- **subquery**
  
  Specifies what to test for. For information on subqueries, see the SELECT statement.

#### Description

Returns true if the `<subquery>` returns a result set that is not empty and returns false if the `<subquery>` returns an empty result set.
4.7.5 IN Predicate

Searches for a value in a set of values and returns true or false.

Syntax

```sql
<in_predicate> ::= <search_for_expression> [ NOT ] IN { <search_in_expression_list> | <subquery> }
```

Syntax Elements

- **search_for_expression**
  The column name or value expression to search for in the set.

- **search_in_expression_list**
  Specifies one or more expressions in which to search for `<search_for_expression>`.

  ```sql
  <search_in_expression_list> ::= <expression> [{, <expression> }...]
  ```

- **subquery**
  Specifies a subquery to search for `<search_for_expression>` in.

- **NOT**
  Inverts the operation of the IN predicate: true is returned if `<search_for_expression>` is not found in the specified set.

Description

True will be returned if the value of `<search_for_expression>` is found in the `<search_in_expression_list>` or `<subquery>.

An expression is either a simple expression such as a character, a date or a number, or it can be a scalar subquery.

Tuples are supported for `<search_for_expression>` and `<search_in_expression_list>`. For example,

```sql
SELECT * FROM "mySchema"."myTable" WHERE (zone, company) IN (('124110','00'), ('124116','00'));
```
Tuples (list of tuples) are also allowed when using subqueries instead of `<search_in_expression>`.

While '=' (equivalent to IN) and '<>' (equivalent to NOT IN) are supported, comparisons using symbols such as '<', '>', and '!' are not supported.

Symbols '=' and '<>' (with a list of tuples) is supported only when using quantified comparison with SOME/ANY/ALL.

**Examples**

```sql
CREATE COLUMN TABLE "my_tab"
( order_nr NVARCHAR(10),
  item_nr  INTEGER,
  some_text NVARCHAR(100),
  PRIMARY KEY (order_nr, item_nr) );

INSERT INTO "my_tab" (order_nr, item_nr, some_text) VALUES ('A000000001', 1, 'A1 First Item');
INSERT INTO "my_tab" (order_nr, item_nr, some_text) VALUES ('A000000001', 2, 'A1 Second Item');
INSERT INTO "my_tab" (order_nr, item_nr, some_text) VALUES ('B000000001', 1, 'B1 First Item');
INSERT INTO "my_tab" (order_nr, item_nr, some_text) VALUES ('A000000002', 1, 'A2 First Item');
INSERT INTO "my_tab" (order_nr, item_nr, some_text) VALUES ('A000000002', 2, 'A2 Second Item');
SELECT * FROM "my_tab"
  WHERE (order_nr, item_nr) = ('A000000001', 2);

SELECT * FROM "my_tab"
  WHERE (order_nr, item_nr) <> ('A000000001', 2);
SELECT * FROM "my_tab"
  WHERE (order_nr, item_nr) != ('A000000001', 2);
-- works only when sub-select returns zero or one row
SELECT * FROM "my_tab"
  WHERE (order_nr, item_nr) != (select order_nr, item_nr from "my_tab" where some_text like 'B1%');
-- error: feature not supported: only '=' and '<>' '/' '!=' operators are allowed here
SELECT * FROM "my_tab"
  WHERE (order_nr, item_nr) > ('A000000001', 2);
-- quantified comparison with SOME/ANY/ALL and sub-select is also supported
SELECT * FROM "my_tab"
  WHERE (order_nr, item_nr) <> ALL (select order_nr, item_nr from "my_tab" where some_text like 'B1%');
-- IN predicate is equivalent to quantified comparison = ANY/SOME
SELECT * FROM "my_tab"
  WHERE (order_nr, item_nr) in (('A000000001', 2), ('B000000001', 1));
SELECT * FROM "my_tab"
  WHERE (order_nr, item_nr) = ANY (('A000000001', 2), ('B000000001', 1));
```

**Related Information**

SELECT Statement (Data Manipulation) [page 1104]

SAP HANA SQL Reference Guide for SAP HANA Platform
4.7.6 LIKE Predicate

Performs a comparison to see if a character string matches a specified pattern.

Syntax

```sql
<like_predicate> ::= <source_expression> [ NOT ] LIKE <pattern_expression>
[ ESCAPE <escape_expression> ]
```

Syntax Elements

- **source_expression** Specifies the column or character string in which to search for `<pattern_expression>`. Specifying NOT inverts the operation of the LIKE predicate.
- **pattern_expression** Specifies the pattern to search for in `<source_expression>`.
- **escape_expression** Specifies the escape character used in comparison string `<pattern_expression>`, if any.

Description

The LIKE predicate performs string comparisons: `<source_expression>` is tested for the pattern contained in `<pattern_expression>`. LIKE returns true if the value of `<pattern_expression>` is found in `<source_expression>` (assuming NOT is not set).

Wildcard characters ( % ) and ( _ ) can be used in `<pattern_expression>`. The percentage sign (%) wildcard matches zero or more characters. The underscore (_) wildcard matches exactly one character.

To match a percentage sign or underscore with the LIKE predicate, an escape character must be placed in front of the wildcard character. Use ESCAPE `<escape_expression>` to specify the escape character you are using. For example, LIKE 'data_%' ESCAPE '_' matches the string data%, and LIKE 'data__%' ESCAPE '_' (that is, two underscores, followed by a percent sign) matches a string that starts with 'data_'.

The underscore (_) and percentage sign ( % ) are ASCII characters.

An expression is either a simple expression such as a character, a date or a number, or it can be a scalar subquery.

If any of `<source_expression>`, `<pattern_expression>`, or `<escape_expression>` is NULL, then the predicate returns UNKNOWN.
4.7.7 LIKE_REGEXPR Predicate

Performs regular expression matching.

Syntax

```
<regex_subject_string> LIKE_REGEXPR <pattern> [ FLAG <flag> ]
```

Syntax Elements

`regex_subject_string`

Specifies the string that the search pattern should be applied to. If `regex_subject_string` is empty, then the result is empty.

```
<regex_subject_string> ::= <string>
```

`pattern`

Specifies a search pattern based on Perl Compatible Regular Expression (PCRE).

```
<pattern> ::= !!Perl Compatible Regular Expression
```

`flag`

Specifies the matching behavior of the predicate.

```
<flag> ::= i | m | s | x | U
```

Flag options

<table>
<thead>
<tr>
<th>Flag option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Enables case-insensitive matching</td>
</tr>
<tr>
<td>m</td>
<td>Enables multiline mode, where the <code>subject_string</code> will be treated as multiple lines and the expression ^ and $ match just after or just before, respectively, a line terminator or the end of the input sequence</td>
</tr>
<tr>
<td>s</td>
<td>Enables the expression <code>.</code> as a wildcard to match any character, including a line terminator</td>
</tr>
</tbody>
</table>
### Description

This predicate performs regular expression matching.

If any of `<pattern>`, `<flag>`, or `<regex_subject_string>` is NULL, then the predicate returns UNKNOWN.

### Example

This fictitious example searches for text like `them` or `this` in the table `mytab`:

```sql
SELECT * FROM mytab WHERE text LIKE_REGEXPR ' them|this ';```

### 4.7.8 MEMBER OF Predicate

Determines whether a value is a member of an array.

### Syntax

```
<expression> [NOT] MEMBER OF <array_value_expression>
```

### Description

Determines whether a value is a member of an array.
Example

The following statements create a table and insert arrays into it:

```
CREATE COLUMN TABLE ARRAY_TEST (IDX INT, VAL INT ARRAY);
INSERT INTO ARRAY_TEST VALUES (1, ARRAY(1, 2, 3));
INSERT INTO ARRAY_TEST VALUES (2, ARRAY(10, 20, 30, 40));
INSERT INTO ARRAY_TEST VALUES (3, ARRAY(10, 20, 30, 40));
INSERT INTO ARRAY_TEST VALUES (4, ARRAY(80, 90, 100, 110));
```

The following statement tests for arrays where 10 is a member, and returns the IDX values for the qualifying arrays.

```
SELECT IDX FROM ARRAY_TEST WHERE 10 MEMBER OF VAL;
```

<table>
<thead>
<tr>
<th>IDX</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

The following statement tests for arrays where 10 is not a member, and returns the IDX values for the qualifying arrays.

```
SELECT IDX FROM ARRAY_TEST WHERE 10 NOT MEMBER OF VAL;
```

<table>
<thead>
<tr>
<th>IDX</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

4.7.9 NULL Predicate

Performs a comparison of the value of an expression with NULL.

Syntax

```
<null_predicate> ::= <expression> IS [NOT] NULL
```

Syntax Elements

- expression
An expression is either a simple expression such as a character, a date or a number, or it can be a scalar subquery.

**IS NULL**

Returns true if the value of `<expression>` is NULL.

**IS NOT NULL**

Returns true if the value of `<expression>` is not NULL.

**Description**

Performs a comparison of the value of an expression with NULL.

**Related Information**

Expressions [page 56]
SELECT Statement (Data Manipulation) [page 1104]

### 4.8 Session Variables

The following table lists predefined session variables and whether they can be set by the user (using a SET statement) or a client (API) call, or whether they are set exclusively by the server.

<table>
<thead>
<tr>
<th>Predefined Variable (M_SESSION_CONTEXT.KEY)</th>
<th>Value Constraint</th>
<th>Set by</th>
<th>In M_SESSION_CONTEXT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION</td>
<td>NVARCHAR(256)</td>
<td>User/Client</td>
<td>Yes</td>
<td>Specifies the application name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Server usage:</strong> Traces, statistical data and trace filters</td>
</tr>
<tr>
<td>Predefined Variable (M_SESSION_CONTEXT.KEY)</td>
<td>Value Constraint</td>
<td>Set by</td>
<td>In M_SESSION_CONTEXT</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>APPLICATIONVERSION</td>
<td>NVARCHAR(256)</td>
<td>User/Client</td>
<td>Yes</td>
<td>Specifies the application version information. Applications can use their “own” version naming, no format is predefined.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Server usage:</strong> Traces, statistical data and trace filters</td>
</tr>
<tr>
<td>APPLICATIONUSER</td>
<td>NVARCHAR(256)</td>
<td>User/Client</td>
<td>Yes</td>
<td>Specifies the application-defined user name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Server usage:</strong> Traces, statistical data and trace filters</td>
</tr>
<tr>
<td>APPLICATIONACTION</td>
<td>NVARCHAR(256)</td>
<td>User/Client</td>
<td>Yes</td>
<td>Specifies that with APPLICATIONACTION the application can define which logical action/step it is currently performing. The usage is up to the application and could be coarse grain like 'create sales order' or fine grain like 'refresh service tab' in SAP HANA Studio.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Server usage:</strong> Traces, statistical data and trace filters</td>
</tr>
<tr>
<td>APPLICATIONCOMPONENT</td>
<td>NVARCHAR(64)</td>
<td>Client</td>
<td>Yes</td>
<td>Specifies the name of the application component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Server usage:</strong> Traces, statistical data and trace filters</td>
</tr>
<tr>
<td>Predefined Variable (M_SESSION_CONTEXT.KEY)</td>
<td>Value Constraint</td>
<td>Set by</td>
<td>In M_SESSION_CONTEXT</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>APPLICATIONCOMPO-</td>
<td>VARCHAR(32)</td>
<td>Client</td>
<td>Yes</td>
<td>Specifies the type of application component. For example, UPD.</td>
</tr>
<tr>
<td>NENTTYPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APPLICATIONSOURCE</td>
<td>NVARCHAR(256)</td>
<td>User/Client</td>
<td>Yes</td>
<td>Specifies that with APPLICATIONSOURCE the application can define from which source file SAP HANA was called.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The usage is up to the application and could be something like one of the following examples:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• <code>&lt;abap_program_name&gt;:&lt;line_number&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• <code>&lt;java_source_file_name&gt;:&lt;line_number&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• <code>&lt;package_name&gt; / &lt;file_name&gt;:&lt;line_number&gt;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Server usage</strong>: Traces, statistical data and trace filters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predefined Variable (M_SESSION_CONTEXT.KEY)</td>
<td>Value Constraint</td>
<td>Set by</td>
<td>In M_SESSION_CONTEXT</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------</td>
<td>--------------</td>
<td>----------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CASE_SENSITIVE</td>
<td>VARCHAR(5)</td>
<td>User/Client</td>
<td>Only if set to FALSE</td>
<td>Specifies that with CASE_SENSITIVE set to TRUE, only the exact match is found. With CASE_SENSITIVE set to FALSE, any upper/lowercase combination of the search string is found. The default is TRUE.</td>
</tr>
</tbody>
</table>

If a case-insensitive search or compare is done outside of a filter on a table, the corresponding columns may be returned in uppercase. This is because an UPPER function is applied by the optimizer to influence the result to achieve a case-insensitive search. For example, ```SELECT * FROM tab WHERE a LIKE '%a%'``` works well, but ```SELECT * FROM tab1 WHERE name LIKE '%case%' UNION SELECT * FROM tab2 WHERE name LIKE '%case%'``` returns the results in uppercase due the required union comparison.

Also, CASE_SENSITIVE is only supported for VARCHAR, NVARCHAR and SHORTTEXT values.
<table>
<thead>
<tr>
<th>Predefined Variable (M_SESSION_CONTEXT.KEY)</th>
<th>Value Constraint</th>
<th>Set by</th>
<th>In M_SESSION_CONTEXT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE_FORMAT</td>
<td>-</td>
<td>User/Client</td>
<td>Yes</td>
<td>Specifies the default DATE format to apply for the session.</td>
</tr>
<tr>
<td>DEBUG_TOKEN</td>
<td>VARCHAR(32)</td>
<td>User/Client</td>
<td>Yes</td>
<td>Specifies that with DEBUG_TOKEN you can set a session variable, used by the debugger, to attach criterion.</td>
</tr>
</tbody>
</table>

**Server usage**: Defines the case-sensitivity during the search.

- WHERE (including LIKE)
- JOIN
- HAVING
- CASE
- GROUP BY
- ORDER BY
- Window functions
- UNION

**Server usage**: M_DEBUG_SESSIONS.ATTACH_FILTER_DEBUG_TOKEN
<table>
<thead>
<tr>
<th>Predefined Variable (M_SESSION_CONTEXT.KEY)</th>
<th>Value Constraint</th>
<th>Set by</th>
<th>In M_SESSION_CONTEXT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE_ABSTRACT_SQL_PLAN_ENTRY</td>
<td>User/Client</td>
<td>Yes</td>
<td></td>
<td>Specifies the ID of the SQL plan that is enabled in the current session. When the query for this plan is executed, a comment is appended to the statement so that it can be identified in the plan cache.</td>
</tr>
<tr>
<td>PROTOCOL_VERSION</td>
<td>Server</td>
<td>Yes</td>
<td></td>
<td>Specifies the protocol version of the client interface libraries, formatted as: <code>&lt;protocol_version&gt;</code> <code>&lt;distribution_protocol_version&gt;, &lt;data_format_version&gt;</code>. For example, 4.1(1.1) &lt;br&gt;&lt;br&gt;&lt;strong&gt;Server usage&lt;/strong&gt;: Internal</td>
</tr>
<tr>
<td>SECONDDATE_FORMAT</td>
<td>User/Client</td>
<td>Yes</td>
<td></td>
<td>Specifies the default SECONDDATE format to apply for the session. &lt;br&gt;&lt;br&gt;&lt;strong&gt;Server usage&lt;/strong&gt;: No</td>
</tr>
<tr>
<td>SPATIAL_OUTPUT_REPRESENTATION</td>
<td>User/Client</td>
<td>Yes</td>
<td></td>
<td>Specifies the output format of ST_Geometry and ST_Point. Valid options are WKB, the default, and EWKB.</td>
</tr>
<tr>
<td>Predefined Variable (M_SESSIONCONTEXT.KEY)</td>
<td>Value Constraint</td>
<td>Set by</td>
<td>In M_SESSIONCONTEXT</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-----------------</td>
<td>--------</td>
<td>---------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>TEMPORAL_SYSTEM_TIME_AS_OF</td>
<td></td>
<td>User/Client</td>
<td>Yes</td>
<td>For use with system-versioned tables. Setting this session variable automatically adds a FOR SYSTEM_TIME AS OF clause to all system-versioned tables contained in subsequent SELECT statements (assuming a FOR SYSTEM_TIME AS OF is not already specified in the SELECT statement).</td>
</tr>
<tr>
<td>TEMPORAL_APPLICATION_TIME_AS_OF</td>
<td></td>
<td>User/Client</td>
<td>Yes</td>
<td>For use with application-time period tables. Setting this session variable automatically adds a FOR APPLICATION_TIME AS OF clause to all application-time period tables contained in subsequent SELECT statements (assuming a FOR APPLICATION_TIME AS OF is not already specified in the SELECT statement).</td>
</tr>
<tr>
<td>TIME_FORMAT</td>
<td></td>
<td>User/Client</td>
<td>Yes</td>
<td>Specifies the default TIME format to apply for the session.</td>
</tr>
<tr>
<td>TIMESTAMP_FORMAT</td>
<td></td>
<td>User/Client</td>
<td>Yes</td>
<td>Specifies the default TIMESTAMP format to apply for the session.</td>
</tr>
<tr>
<td>Predefined Variable (M_SESSIONCONTEXT.KEY)</td>
<td>Value Constraint</td>
<td>Set by</td>
<td>In M_SESSIONCONTEXT</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------</td>
<td>--------</td>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>TOTAL_ROWCOUNT</td>
<td>Integer</td>
<td>Server</td>
<td>No, only accessible from: <code>SELECT session_context.t('TOTAL_ROWCOUNT') FROM DUMMY.</code></td>
<td>Specifies the total row count. Specifying <code>SELECT ... LIMIT x</code> returns up to <code>x</code> rows. With the optional <code>TOTAL ROWCOUNT</code> the (estimated) total number of rows is set in <code>TOTAL_ROWCOUNT</code> variable. This variable is not deleted/reset by non <code>TOTAL ROWCOUNT</code> statements. The similar <code>SELECT TOP x ...</code> does not support this extension. This feature is only supported for column tables and column views. It can return exact or estimated row count or only the given <code>LIMIT</code>. The concrete return value depends on the statement, used tables and views and internally chosen SQL optimizer strategy. Usage is recommended only for interactive paged searches to show information like &quot;result 11 to 20 of estimated 50000&quot;. <strong>Server usage:</strong> <code>SELECT ... LIMIT x TOTAL ROWCOUNT</code></td>
</tr>
</tbody>
</table>
### Predefined Variable (M_SESSIONCONTEXT.KEY)

<table>
<thead>
<tr>
<th>Predefined Variable</th>
<th>Value Constraint</th>
<th>Set by</th>
<th>In M_SESSIONCONTEXT</th>
<th>Description</th>
</tr>
</thead>
</table>
| TRACEPROFILE        | -                | User/Client     | Yes                 | Specifies the name of the trace profile. This is used for manual activation of trace profiles defined in inifiles as `[traceprofile_<name>]`.

The value can contain one traceprofile name or a comma separated list of traceprofile names.

In SAP HANA Studio, the traceprofiles are accessible as End-To-End Traces.

**Server usage:** Trace filter

### Related Information

- **M_SESSION_CONTEXT System View** [page 2133]
- **SET [SESSION] Statement (Session Management)** [page 1171]
- **UNSET [SESSION] Statement (Session Management)** [page 1183]

### 4.9 SQL Functions

Documents the built-in SQL functions that are provided with SAP HANA.

Analytic functions are positioned in this guide as aggregate functions and window aggregate functions.

- **Alphabetical List Of Functions** [page 84]
- **Aggregate Functions** [page 414]
  
  Aggregate functions are analytic functions that calculate an aggregate value based on a group of rows.

- **Array Functions** [page 415]
  
  Array functions take arrays as input.

- **Data Type Conversion Functions** [page 416]
  
  Data type conversion functions convert data from one data type to another data type.
Datetime Functions [page 417]
Date and time functions perform operations on date and time data types or return date or time information.

Fulltext Functions [page 418]
Fulltext functions perform operations on data that has a fulltext index.

Hierarchy Functions [page 418]
Hierarchy functions allow you to work with hierarchical data such as tables with rows arranged in a tree or directed graph.

JSON Functions [page 419]
JSON functions are functions that return or operate on JSON data.

Miscellaneous Functions [page 420]
SAP HANA supports many functions that return system values and perform various operations on values, expressions, and return values of other functions.

Numeric Functions [page 421]
Numeric functions perform mathematical operations on numerical data types or return numeric information.

Security Functions [page 422]
Security functions provide special functionality for security purposes.

Series Data Functions [page 422]
Series data functions provide special functionality for series data and series tables.

String Functions [page 423]
String functions perform extraction and manipulation on strings, or return information about strings.

Window Functions and the Window Specification [page 424]
Window functions allow you to perform analytic operations over a set of input rows.

4.9.1 Alphabetical List Of Functions

4.9.1.1 ABAP_ALPHANUM Function (String)

Converts a string to what would result if the string was transformed into an ALPHANUM type and then converted back to a string.

Syntax

\[
\text{ABAP\_ALPHANUM}(\ <\text{string}>\ ,\ <\text{chars}>\ )
\]
**Syntax Elements**

- **string** Specifies the input string to convert.
- **chars** Specifies the length to left-pad `<string>` with (using 0s) if `<string>` is shorter than the length specified by `<chars>`.

**Description**

The input value, `<string>`, is checked to find out if it is numeric or a mixture of numeric and non-numeric. If `<string>` is numeric, then it is left-padded with zeroes up to the length specified by `<chars>`, if necessary. If `<string>` is a mixture or numeric and non-numeric, or is empty, then it is returned unchanged.

Strings that start with spaces and contain only digits after the whitespace prefix are regarded as numeric.

**Example**

The following example returns 012. Since `<chars>` is set to 3, the left-most digit is padded with a 0.

```sql
SELECT ABAP_ALPHANUM('12', 3) FROM DUMMY;
```

The following example returns 1234. The input value was numeric and was not shorter than `<chars>`, so no left-padding was required.

```sql
SELECT ABAP_ALPHANUM('1234', 3) FROM DUMMY;
```

The following example returns 12x. Because the input value contained a mixture of numeric and non-numeric, it is returned unchanged.

```sql
SELECT ABAP_ALPHANUM('12x', 13) FROM DUMMY;
```

**Related Information**

[Character String Data Types](page 36)
4.9.1.2 ABAP_DF16RAW_TO_SMALLDECIIMAL Function (String)

Converts from ABAP decfloat type DF16RAW to HANA SMALLDECIIMAL.

Syntax

ABAP_DF16RAW_TO_SMALLDECIIMAL(<binary_data_in_df16raw_format>)

Syntax Elements

binary_data_in_df16raw_format The format to convert: Decimal Floating Point, 16 Digits, RAW on database

Description

Converts from ABAP decfloat type DF34_RAW to HANA decfloat type decimal.

Example

The following example converts the binary value to type decimal.

```
CREATE TABLE t1(b binary(16))
INSERT INTO t1 values(X'BF2C000000000000')
INSERT INTO t1 values(X'BF54000000000001')
INSERT INTO t1 values(X'3DE3E7F9F9FE7F9F')
SELECT ABAP_DF16RAW_TO_SMALLDECIIMAL(b) FROM t1
/* returns
   [ Decimal('0.1')
   , Decimal('1.000000000000001')
   , Decimal('-0.1')]
*/
```
4.9.1.3 ABAP_DF34RAW_TO_DECIMAL Function (String)

Converts from ABAP decfloat type DF34_RAW to HANA decfloat type decimal.

Syntax

ABAP_DF34RAW_TO_DECIMAL(<binary_data_in_df34raw_format>)

Syntax Elements

binary_data_in_df34raw_format The format to convert: Decimal Floating Point, 34 Digits, RAW on database

Description

Converts from ABAP decfloat type DF34_RAW to HANA decfloat type decimal.

Example

The following example converts the binary value to type decimal.

```sql
CREATE TABLE t2 (b binary(64));
INSERT INTO t2 values(X'BDDBC0000000000000000000000000000');
INSERT INTO t2 values(X'BDBE4000000000000000000000000001');
INSERT INTO t2 values(X'BDBEC8D9428D937193B9CEA0D7F45DF7');
SELECT ABAP_DF34RAW_TO_DECIMAL(b) FROM t2
/* returns
[Decimal('0.1'), Decimal('1.00000000000000000000000000000001'),
Decimal('3.141592653589793238462643383279503')] */
```
4.9.1.4 ABAP_NUMC Function (String)

Converts an input string to a string of a specified length, that contains only digits.

Syntax

ABAP_NUMC( <string>, <length> )

Syntax Elements

- **string** Specifies the input string to convert.
- **length** Specifies the length that the returned string should be.

Description

String and numeric values are left-padded with 0s if `<string>` is shorter than `<length>`. If `<string>` is longer than `<length>`, then characters are truncated from the left.

Numeric values are rounded to their integer part and the sign is discarded.

TIMESTAMP, SECONDDATE, DATE, and TIME values are padded and truncated from the right side, as necessary.

DECIMAL and SMALLDECIMAL inputs are never truncated but create an overflow error if they are found to be too long.

Example

The following example returns 012.

```
SELECT ABAP_NUMC(12, 3) FROM DUMMY;
```

The following example returns 234.

```
SELECT ABAP_NUMC(1234, 3) FROM DUMMY;
```

The following example returns 0123000000000000000000.

```
SELECT ABAP_NUMC(1.23e18, 20) FROM DUMMY;
```
The following example returns `001235`.

```
SELECT ABAP_NUMC(-1234.5, 6) FROM DUMMY;
```

**Related Information**

Character String Data Types [page 36]

### 4.9.1.5 ABAP_LOWER Function (String)

Converts all characters in a specified string to lowercase.

**Syntax**

```
ABAP_LOWER(<string>)
```

**Syntax Elements**

- **string** The string to convert to lowercase.

**Description**

Converts all characters in the `<string>` parameter to lowercase.

**Example**

The following example converts the given string to lowercase and returns the value `ant`:

```
SELECT ABAP_LOWER ('AnT') "lower" FROM DUMMY;
```
4.9.1.6 ABAP_UPPER Function (String)

Converts all characters in the specified string to uppercase.

Syntax

```
ABAP_UPPER(<string>)
```

Syntax Elements

- `string` The string to convert to uppercase.

Description

Converts all characters in the `<string>` parameter to uppercase.

Example

The following example converts the given string to uppercase and returns the value ANT:

```
SELECT ABAP_UPPER ('Ant') "uppercase" FROM DUMMY;
```
4.9.1.7   ABS Function (Numeric)

Returns the absolute value of a numeric argument.

**Syntax**

```plaintext
ABS (<num>)
```

**Syntax Elements**

- `<num>` Specifies the numeric argument.

**Description**

Returns the absolute value of the numeric argument `<num>`.

**Example**

The following example returns the value 1 for "absolute":

```sql
SELECT ABS (-1) "absolute" FROM DUMMY;
```

**Related Information**

[Numeric Data Types](page 48)
4.9.1.8 ACOS Function (Numeric)

Returns the arc-cosine, in radians, of a numeric argument between -1 and 1.

Syntax

\[
\text{ACOS} \left( \text{num} \right)
\]

Syntax Elements

\text{num} \quad \text{Specifies a numeric argument between -1 and 1.}

Description

Returns the arc-cosine, in radians, of the numeric argument <n> between -1 and 1.

Example

The following example returns the value 1.0471975511965979:

```
SELECT ACOS (0.5) "acos" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]
4.9.1.9 ADD_DAYS Function (Datetime)

Computes the specified date, plus or minus the specified number of days.

Syntax

```
ADD_DAYS(<date>, <num_days>)
```

Syntax Elements

- **date** Specifies the date to increment.
- **num_days** Specifies the number of days (integer). A positive integer increments the date by the specified amount; a negative integer decrements the date.

Description

Computes the specified date plus or minus the specified number of days.

Example

The following example increments the date value 2009-12-05 by 30 days and returns the value 2010-01-04:

```
SELECT ADD_DAYS (TO_DATE ('2009-12-05', 'YYYY-MM-DD'), 30) "add days" FROM DUMMY;
```

The following example decrements the date value 2009-12-05 by 30 days and returns the value 2009-11-05:

```
SELECT ADD_DAYS (TO_DATE ('2009-12-05', 'YYYY-MM-DD'), -30) "subtract days" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]
4.9.1.10 ADD_MONTHS Function (Datetime)

Computes the specified date plus the specified number of months.

Syntax

```sql
ADD_MONTHS(<date>, <num_months>)
```

Syntax Elements

- **date**: The date that is incremented.
- **num_months**: The number of months by which to increment the date.

Description

Computes the specified date plus the specified number of months.

To compute the date so that the output date is set to the last day of the month when the input date is the last day of the month, use the ADD_MONTHS_LAST function.

The parameter `<date>` must be implicitly or explicitly converted to one of the following SQL data types:

- **DATE**
- **TIMESTAMP**
- **SECONDDATE**

The SQL data type of the output parameters is the same as the SQL data type of the input parameters. For example, ADD_MONTHS_LAST(DATE) returns a date, while ADD_MONTHS_LAST(TIMESTAMP) returns a timestamp.

Example

The following example increments the date 2009-12-05 by one month, and returns the value 2010-01-05:

```sql
SELECT ADD_MONTHS (TO_DATE ('2009-12-05', 'YYYY-MM-DD'), 1) "add months" FROM DUMMY;
```
4.9.1.11 ADD_MONTHS_LAST Function (Datetime)

Computes the specified date plus the specified number of months, with the output date being the last day of the month if the input date is the last day of the month, even if those two dates differ.

Syntax

```
ADD_MONTHS_LAST(<date>, <num_months>)
```

Syntax Elements

- **date** The date that is incremented.
- **num_months** The number of months by which to increment the date.

Description

Computes the specified date plus the specified number of months. If the input date is the last day of the input month, then the output date is set to the last day of the output month.

To compute the date plus months without using the last day of the month functionality, use the ADD_MONTHS function.

The parameter `<date>` must be implicitly or explicitly converted to one of the following SQL data types:

- DATE
- TIMESTAMP
- SECONDDATE

The SQL data type of the output parameters is the same as the SQL data type of the input parameters. For example, `ADD_MONTHS_LAST(DATE)` returns a date, while `ADD_MONTHS_LAST(TIMESTAMP)` returns a timestamp.
Example

The following example increments the date 2009-02-28 (the last day in February), by one month, and returns the value 2009-03-31, (the last day of March).

```
SELECT ADD_MONTHS_LAST (TO_DATE ('2009-02-28', 'YYYY-MM-DD'), 1) "add months last" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]

4.9.1.12 ADD_NANO100 Function (Datetime)

Adds the specified number of 10^-7 unit of seconds to the specified TIMESTAMP value.

Syntax

```
ADD_NANO100( <time>, <num> )
```

Syntax Elements

- **time** The TIMESTAMP value to add the units of seconds to.
- **num** The number of 10^-7 unit of seconds to increment the TIMESTAMP value by.

Description

Computes the specified TIMESTAMP value plus the specified number of microseconds.
**Example**

The following example increments the TIMESTAMP value by 864000000000 microseconds and returns 1000-01-02 10:00:00.0000000:

```sql
SELECT ADD_NANO100(TO_TIMESTAMP('1000-01-01 10:00:00.0000000'), 864000000000) FROM DUMMY;
```

The following example increments the TIMESTAMP value by 1 microsecond and returns 1000-01-01 10:00:00.0000001:

```sql
SELECT ADD_NANO100(TO_TIMESTAMP('1000-01-01 10:00:00.0000000'), 1) FROM DUMMY;
```

**Related Information**

ADD_SECONDS Function (Datetime) [page 97]

**4.9.1.13  ADD_SECONDS Function (Datetime)**

Computes the specified time plus the specified seconds.

**Syntax**

```sql
ADD_SECONDS(<time>, <num_seconds>)
```

**Syntax Elements**

- **time** The time that is incremented.
- **num_seconds** The number of seconds to increment the time by. Fractional values are supported for milliseconds and microseconds:
  - 1/1000 – milliseconds
  - 1/1000000 - microseconds
Description

Computes the specified time plus the number of specified seconds.

Example

The example increments the TIMESTAMP 2012-01-01 23:30:45 by 60*30 seconds, and returns the value 2012-01-02 00:00:45.0:

```
SELECT ADD_SECONDS (TO_TIMESTAMP ('2012-01-01 23:30:45'), 60*30) "add seconds"
FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]
ADD_NANO100 Function (Datetime) [page 96]

4.9.1.14 ADD_WORKDAYS Function (Datetime)

Computes a date by adding a number of workdays to a starting date.

Syntax

```
ADD_WORKDAYS{
  <factory_calendar_id>,
  <start_date>,
  <workdays>
  [, <source_schema>]
  [, <table_name_suffix>],
  [, <client>] ] ] ]
```

Syntax Elements

- **factory_calendar_id**
  
  Specifies the ID of the factory calendar.
  
  ```
  <factory_calendar_id> ::= <string_literal>
  ```
**start_date**

Specifies the start date that the work days will be added to. You can use either a DATE type or a date format string (for example ‘20140101’ or ‘2014-01-01’) for this parameter.

```plaintext
<start_date> ::= <string_literal> | <date>
```

**workdays**

Specifies the number of working days to be added to the starting date.

```plaintext
<workdays> ::= <signed_integer>
```

When `<workdays>` is positive, the resulting calculated date is the next working day after the period defined by the number of `<workdays>`. When `<workdays>` is negative, the resulting calculated date is the previous working day before the period defined by the number of `<workdays>`.

**source_schema**

Specifies the schema containing the Factory Calendar table or the Factory And Holiday Calendar tables.

```plaintext
<source_schema> ::= <string_literal>
```

This parameter can be omitted if the schema is the same as the current schema.

**table_name_suffix**

Specifies the suffix appended to the standard Factory Calendar table name or the standard Factory And Holiday Calendar table names if using an alternative set of tables. For example, with suffix ‘_XYZ’ the function accesses the alternative tables TFACS_XYZ or FHC_C_FCAL_XYZ instead of standard tables TFACS or FHC_C_FCAL. If omitted or empty, the standard tables are used.

```plaintext
<table_name_suffix> ::= <string_literal>
```

**client**

Specifies the client used for the client-dependent Factory And Holiday Calendar implementation. If omitted or empty, then the function tries to retrieve the client from the context (user parameter), which can be set in the context by ALTER USER `<user_name>` SET PARAMETER CLIENT = `<client>`. The client is ignored if the client-independent Factory Calendar implementation is active.

```plaintext
<client> ::= <string_literal>
```

**Return Type**

**DATE**

**Examples**

The following examples use the Factory Calendar table for the month of January 2014 exclusively. For examples using the Factory And Holiday Calendar, see Factory Calendar.
<table>
<thead>
<tr>
<th>TFACS bitfield</th>
<th>Day of the month</th>
<th>Reason for not working</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Public Holiday</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>Weekend</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>Weekend</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>Public Holiday</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>Weekend</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>Weekend</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>18</td>
<td>Weekend</td>
</tr>
<tr>
<td>0</td>
<td>19</td>
<td>Weekend</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>25</td>
<td>Weekend</td>
</tr>
<tr>
<td>0</td>
<td>26</td>
<td>Weekend</td>
</tr>
</tbody>
</table>
The following example returns the value 10.01.2014:

```sql
SELECT ADD_WORKDAYS('01', '2014-01-09', 1, 'FCTEST') "result date" FROM DUMMY;
```

From this result you can see that a single workday was added to the start date of the ninth, with the result being the tenth.

The following example returns the value 09.01.2014:

```sql
SELECT ADD_WORKDAYS('01', '2014-01-10', -1, 'FCTEST') "result date" FROM DUMMY;
```

From this result you can see that the tenth was considered to be a working day, producing a final result of the ninth.

The following example takes positive workday input and returns the value 20.01.2014, a result after a non-working weekend period:

```sql
SELECT ADD_WORKDAYS('01', '2014-01-17', 1, 'FCTEST') "result date" FROM DUMMY;
```

From this result you can see that a single workday was added to the start date of the 17th. This produced the resulting day of the 20th, which allowed for the non-working period of the weekend.

The following example takes negative workday input and returns the value 17.01.2014, showing a result after a non-working weekend period:

```sql
SELECT ADD_WORKDAYS('01', '2014-01-20', -1, 'FCTEST') "result date" FROM DUMMY;
```

From this result you can see that the 20th was considered to be the working day, producing a result of the 17th. This result takes into account the non-working time of the weekend.

The following complex example returns the value 27.01.2014:

```sql
SELECT ADD_WORKDAYS('01', '2014-01-01', 16, 'FCTEST') "result date" FROM DUMMY;
```

The statement above adds 16 working days to the start date of the first. The system takes into account the weekends (4th, 5th, 11th, 12th, 18th and 19th) and public holidays (1st and 6th) during the working period, which gives the last working day as the 24th. The system then returns the next possible working day after this, which is the 27th.

The following example uses a table for input and returns the values 4.02.2014, 14.05.2014, 05.08.2014, and 30.10.2014:

```sql
CREATE SCHEMA SAPABC;
```
SET SCHEMA "SAPABC";
CREATE ROW TABLE MY_DATES (FCID NVARCHAR(2), STARTDATE DATE, DURATION INTEGER);
INSERT INTO MY_DATES VALUES ('01', '2014-01-01', 30);
INSERT INTO MY_DATES VALUES ('01', '2014-04-01', 28);
INSERT INTO MY_DATES VALUES ('01', '2014-07-01', 25);
INSERT INTO MY_DATES VALUES ('01', '2014-10-01', 20);
SELECT ADD_WORKDAYS(FCID, STARTDATE, DURATION) "shipment date" FROM MY_DATES;

Related Information

Datetime Data Types [page 40]
WORKDAYS_BETWEEN Function (Datetime) [page 400]
ALTER USER Statement (Access Control) [page 654]

4.9.1.15 ADD_YEARS Function (Datetime)

Computes the specified date plus the specified years.

Syntax

ADD_YEARS(<date>, <num_years>)

Syntax Elements

date The date that is incremented.
num_years The number of years by which to increment the date.

Description

Computes the specified date plus the specified number of years.
**Example**

The following example increments the specified date value 2009-12-05 by 1 year, and returns the value 2010-12-05:

```
SELECT ADD_YEARS (TO_DATE ('2009-12-05', 'YYYY-MM-DD'), 1) "add years" FROM DUMMY;
```

**Related Information**

Datetime Data Types [page 40]

**4.9.1.16 ALLOW_PRECISION LOSS Function (Miscellaneous)**

Allows loss of precision when aggregating decimal values using an aggregate expression.

**Syntax**

```
ALLOW_PRECISION_LOSS( <aggregate_expression> )
```

**Syntax Elements**

**aggregate_expression**

The aggregate expression for which to allow precision loss. Currently SUM is the only supported aggregate expression.

```
<aggregate_expression> ::= SUM ( <expression> )
```

**Description**

Use this function to improve performance of an aggregation expression on DECIMAL values when loss of precision is acceptable and to improve performance of the aggregation.
Example

The following example shows the difference in values returned when using the ALLOW_PRECISION_LOSS function:

```
CREATE TABLE TESTTABLE (COL1 decimal(10,5), COL2 decimal);
INSERT INTO TESTTABLE VALUES(1.139999, 1.138888888);
INSERT INTO TESTTABLE VALUES(2.119999, 2.118888888);
INSERT INTO TESTTABLE VALUES(2.119999, 2.118888888);
INSERT INTO TESTTABLE VALUES(2.669999, 2.668888888);
-- The following query, which does not allow precision loss, returns 8.01, 8.01
SELECT SUM(TO_DECIMAL(COL1,10,2)), SUM(TO_DECIMAL(COL2,10,2)) FROM TESTTABLE;
-- The following query, which uses the ALLOW_PRECISION_LOSS function to allow
precision loss, returns 8.04, 8.04
SELECT ALLOW_PRECISION_LOSS(SUM(TO_DECIMAL(COL1,10,2))),
ALLOW_PRECISION_LOSS(SUM(TO_DECIMAL(COL2,10,2))) FROM TESTTABLE;
```

Related Information

Expressions [page 56]

4.9.1.17 ASCII Function (String)

Returns the integer ASCII value of the first character in a specified string.

Syntax

```
ASCII(<string>)
```

Syntax Elements

- **string** Specifies the string to return the integer ASCII value from.

Description

Returns the integer ASCII value of the first character in a string `<string>`.
Example

This example converts the first character of the string Ant into a numeric ASCII value and returns the value 65:

```
SELECT ASCII('Ant') "ascii" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]

4.9.1.18 ASIN Function (Numeric)

Returns the arc-sine, in radians, of a numeric argument.

Syntax

```
ASIN(<number>)
```

Syntax Elements

- number Specifies a numeric argument between -1 and 1.

Description

Returns the arc-sine, in radians, of the numeric argument <number> between -1 and 1.

Example

The following example returns the value 0.5235987755982989 for "asin":

```
SELECT ASIN (0.5) "asin" FROM DUMMY;
```
4.9.1.19 ATAN Function (Numeric)

Returns the arc-tangent, in radians, of a numeric argument.

Syntax

```
ATAN(<number>)
```

Syntax Elements

- `number` Specifies a numeric argument.

Description

Returns the arc-tangent, in radians, of the numeric argument `<number>`. The range of `<number>` is unlimited.

Example

The following example returns the value 0.4636476090008061 for "atan":

```
SELECT ATAN (0.5) "atan" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]
4.9.1.20 ATAN2 Function (Numeric)

Returns the arc-tangent, in radians, of the ratio of two numbers.

Syntax

\[
\text{ATAN2}(<\text{number1}>, <\text{number2}>)
\]

Syntax Elements

- \text{number1} Specifies the first numeric argument.
- \text{number2} Specifies the second numeric argument.

Description

Returns the arc-tangent, in radians, of the ratio of two numbers \text{<number1>} and \text{<number2>}.

Example

The following example returns the value 0.4636476090008061 for "atan2":

```
SELECT ATAN2 (1.0, 2.0) "atan2" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]
4.9.1.21 AUTO_CORR Function (Aggregate)

Computes all autocorrelation coefficients for a given input expression and returns an array of values.

Syntax

\[
\text{AUTO_CORR( } \langle \text{expression} \rangle , \langle \text{maxTimeLag} \rangle \{ \langle \text{series_order_by_clause} \rangle | \langle \text{order_by_clause} \rangle \} )
\]

Syntax Elements

- **expression** Specifies the input expression. \langle \text{expression} \rangle values can be of any numeric type.
- **maxTimeLag** The time frame size is limited by the maxTimeLag parameter. This parameter must be a positive integer. The result size is the minimum of maxTimeLag and column size - 2 for dense series data.
- **series_orderby** \langle \text{series_orderby} \rangle can only be used with an equidistant series.

\[
\langle \text{series_orderby} \rangle ::= \text{SERIES( } \langle \text{series_period} \rangle \langle \text{series_equidistant_definition} \rangle \text{ )}
\]

The use of SQLScript variables in this clause is not supported. The series * must not contain missing elements. For more information about this clause, see the SERIES_GENERATE function.
- **order_by_clause** Specifies the sort order of the input rows.

\[
\langle \text{order_by_clause} \rangle ::= \text{ORDER BY} \langle \text{order_by_expression} \rangle [, \langle \text{order_by_expression} \rangle [...] ]
\]

\[
\langle \text{order_by_expression} \rangle ::= \langle \text{column_name} \rangle [ \text{ASC} | \text{DESC} ] [ \text{NULLS FIRST} | \text{NULLS LAST} ] [ \langle \text{collate_clause} \rangle ]
\]

\[
\langle \text{collate_clause} \rangle ::= \text{COLLATE} \langle \text{collation_name} \rangle
\]

<collate_clause> specifies the collation to use for ordering values in the results. <collate_clause> can only be used on columns defined as NVARCHAR or VARCHAR.<collation_name> is one of the supported collation names listed in the COLLATIONS system view.

Description

Computes all autocorrelation coefficients for a given input expression and returns an array of values.

Pairs that contain at least one null are removed. Even though AUTO_CORR can handle null input values, it is highly recommended to replace null values first (e.g. by using LINEAR_APPROX), which allows for much faster processing.
The output is empty if there are fewer than two rows.

Examples

The example below shows autocorrelation of dense series data and returns
[0.285714, -0.351351, -0.5625, -0.25, 1, 1, 1, 1, 1].

```
CREATE COLUMN TABLE correlationTable (TS_ID VARCHAR(10), DATE DAYDATE, VALUE DOUBLE);
INSERT INTO correlationTable VALUES ('A', '2014-10-01', 1);
INSERT INTO correlationTable VALUES ('A', '2014-10-02', 2);
INSERT INTO correlationTable VALUES ('A', '2014-10-03', 3);
INSERT INTO correlationTable VALUES ('A', '2014-10-04', 4);
INSERT INTO correlationTable VALUES ('A', '2014-10-05', 5);
INSERT INTO correlationTable VALUES ('A', '2014-10-06', 1);
INSERT INTO correlationTable VALUES ('A', '2014-10-07', 2);
INSERT INTO correlationTable VALUES ('A', '2014-10-08', 3);
INSERT INTO correlationTable VALUES ('A', '2014-10-09', 4);
INSERT INTO correlationTable VALUES ('A', '2014-10-10', 5);
SELECT TS_ID, AUTO_CORR(VALUE, 8 SERIES (PERIOD FOR SERIES(DATE)
EQUIDISTANT INCREMENT BY INTERVAL 1 DAY MISSING ELEMENTS NOT ALLOWED))
FROM correlationTable
GROUP BY TS_ID ORDER BY TS_ID;
```

The example below shows autocorrelation of sparse series data without considering missing entries, and returns [1, 1, 1, 1, 1].

```
CREATE COLUMN TABLE correlationTable (ts_id VARCHAR(20), date DAYDATE, val DOUBLE);
INSERT INTO correlationTable VALUES ('A', '2014-10-01', 1);
INSERT INTO correlationTable VALUES ('A', '2014-10-02', 2);
INSERT INTO correlationTable VALUES ('A', '2014-10-04', 3);
INSERT INTO correlationTable VALUES ('A', '2014-10-07', 4);
INSERT INTO correlationTable VALUES ('A', '2014-10-11', 5);
INSERT INTO correlationTable VALUES ('A', '2014-10-21', 6);
INSERT INTO correlationTable VALUES ('A', '2014-10-22', 7);
SELECT ts_id, AUTO_CORR(val, 999 SERIES (PERIOD FOR SERIES(DATE)
EQUIDISTANT INCREMENT BY INTERVAL 1 DAY MISSING ELEMENTS NOT ALLOWED))
FROM correlationTable
GROUP BY ts_id ORDER BY ts_id;
```

The correlationTable has missing entries, such as '2014-10-03', but the WITHIN GROUP clause considers the series to be equidistant with one day intervals, where missing elements are not allowed.

Since the series data is assumed to be dense, the autocorrelation of the data set [1..7] is calculated.

The example below shows autocorrelation of sparse series data considering the missing entries, and returns
[1.0, null, 1.0, null, null, null, null, null, null, null, 1.0].

```
CREATE COLUMN TABLE correlationTable (ts_id VARCHAR(20), date DAYDATE, val DOUBLE);
INSERT INTO correlationTable VALUES ('A', '2014-10-01', 1);
INSERT INTO correlationTable VALUES ('A', '2014-10-02', 2);
INSERT INTO correlationTable VALUES ('A', '2014-10-04', 3);
INSERT INTO correlationTable VALUES ('A', '2014-10-07', 4);
INSERT INTO correlationTable VALUES ('A', '2014-10-11', 5);
INSERT INTO correlationTable VALUES ('A', '2014-10-21', 6);
INSERT INTO correlationTable VALUES ('A', '2014-10-22', 7);
SELECT ts_id, AUTO_CORR(val, 999 SERIES (PERIOD FOR SERIES(DATE)
EQUIDISTANT INCREMENT BY INTERVAL 1 DAY MISSING ELEMENTS NOT ALLOWED))
FROM correlationTable
GROUP BY ts_id ORDER BY ts_id;
```
SELECT ts_id, AUTO_CORR(val, 999 SERIES (PERIOD FOR SERIES(DATE) EQUIDISTANT INCREMENT BY INTERVAL 1 DAY MISSING ELEMENTS ALLOWED)) FROM correlationTable GROUP BY ts_id ORDER BY ts_id;

Autocorrelation works as if there were nulls instead of the missing elements in the column.

**Related Information**

- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- Aggregate Functions [page 414]
- CORR Function (Aggregate) [page 145]
- CORR_SPEARMAN Function (Aggregate) [page 148]
- CROSS_CORR Function (Aggregate) [page 157]
- Expressions [page 56]

**4.9.1.22 AVG Function (Aggregate)**

Returns the arithmetical mean of the expression. This function can also be used as a window function.

**Syntax**

**Aggregate function:**

```
AVG( [ ALL | DISTINCT ] <expression> )
```

**Window function:**

```
AVG( <expression> ) <window_specification>
```

**Syntax Elements**

- **expression**
  Specifies the input data for the function.
- **window_specification**
  Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].
Description

Result type based on input

<table>
<thead>
<tr>
<th>TINYINT</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>BIGINT</th>
<th>DECIMAL (p, s)</th>
<th>DECIMAL</th>
<th>REAL</th>
<th>DOUBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECIMAL(9,6)</td>
<td>DECIMAL(1,6)</td>
<td>DECIMAL(2,6)</td>
<td>DECIMAL(p, s)</td>
<td>DECIMAL</td>
<td>REAL</td>
<td>DOUBLE</td>
<td></td>
</tr>
</tbody>
</table>

Example

The following statements create a table with data for example purposes:

```sql
DROP TABLE "MyProducts";
CREATE TABLE "MyProducts"
(
"Product_ID" VARCHAR(10),
"Product_Name" VARCHAR(100),
"Category" VARCHAR(100),
"Quantity" INTEGER,
"Price" DECIMAL(10,2),
PRIMARY KEY ("Product_ID")
);

INSERT INTO "MyProducts" VALUES('P1','Shirts', 'Clothes', 32, 20.99);
INSERT INTO "MyProducts" VALUES('P2','Jackets', 'Clothes', 16, 99.49);
INSERT INTO "MyProducts" VALUES('P3','Trousers', 'Clothes', 30, 32.99);
INSERT INTO "MyProducts" VALUES('P4','Coats', 'Clothes', 5, 129.99);
INSERT INTO "MyProducts" VALUES('P5','Purse', 'Accessories', 3, 89.49);

The following example averages the prices of all products in the MyProducts table and returns the value 74.50:

```sql
SELECT AVG("Price") FROM "MyProducts";
```

Related Information

Window Functions and the Window Specification [page 424]
Window Aggregate Functions [page 428]
Aggregate Functions [page 414]
Expressions [page 56]
4.9.1.23  BINNING Function (Window)

Partitions an input set into disjoint subsets by assigning a bin number to each row.

Syntax

```
BINNING( <binning_param> => <expression> [ {, <binning_parameter> => <expression> } ... ] ) <window_specification>
```

```
<binning_param> ::= VALUE | BIN_COUNT | BIN_WIDTH | TILE_COUNT | STDDEV_COUNT
```

Syntax Elements

**binning_param**

VALUE is always required. It specifies the column that binning is applied to. When BIN_WIDTH is used, the input column must have a numeric data type. BIN_COUNT specifies the number of equal-width bins. BIN_WIDTH specifies the width of the bins. TILE_COUNT specifies the number of bins with equal number of records. STDDEV_COUNT specifies the number of standard deviations left and right from the mean.

The appropriate binning method is selected based on the parameter specified – exactly one of the last four parameters must be non-NULL. The value assigned to binning method parameter must be an integer expression.

**window_specification**

Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].

Description

The OVER clause must not specify a `<window_order_by_clause>` nor any window frame since the binning function works on an entire partition.

Examples

The following example distributes the value of the input set into four bins that all have an equal width.

```
DROP TABLE weather;
CREATE ROW TABLE weather (station INT, ts DATE, temperature FLOAT);
INSERT INTO weather VALUES(1, '2014-01-01', 0);
INSERT INTO weather VALUES(1, '2014-01-02', 3);
INSERT INTO weather VALUES(1, '2014-01-03', 4.5);
INSERT INTO weather VALUES(1, '2014-01-04', 6);
INSERT INTO weather VALUES(1, '2014-01-05', 6.3);
```
```sql
INSERT INTO weather VALUES(1, '2014-01-06', 5.9);
INSERT INTO weather VALUES(1, '2015-01-01', 1);
INSERT INTO weather VALUES(1, '2015-01-02', 3.4);
INSERT INTO weather VALUES(1, '2015-01-03', 5);
INSERT INTO weather VALUES(1, '2015-01-04', 6.7);
INSERT INTO weather VALUES(1, '2015-01-05', 4.6);
INSERT INTO weather VALUES(1, '2015-01-06', 6.9);
SELECT *, BINNING(VALUE => temperature, BIN_COUNT => 4) OVER () AS bin_num FROM weather;
```

<table>
<thead>
<tr>
<th>STATION</th>
<th>TS</th>
<th>TEMPERATURE</th>
<th>BIN_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01.01.2014</td>
<td>0.0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>02.01.2014</td>
<td>3.0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>03.01.2014</td>
<td>4.5</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>04.01.2014</td>
<td>6.0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>05.01.2014</td>
<td>6.3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>06.01.2014</td>
<td>5.9</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>01.01.2014</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>02.01.2014</td>
<td>3.4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>03.01.2014</td>
<td>5.0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>04.01.2014</td>
<td>6.7</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>05.01.2014</td>
<td>4.6</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>06.01.2014</td>
<td>6.9</td>
<td>4</td>
</tr>
</tbody>
</table>

Related Information

Expressions [page 56]
Window Functions and the Window Specification [page 424]
4.9.1.24 BINTOHEX Function (String)

Converts a binary string to an NVARCHAR hexadecimal value.

Syntax

\[ \text{BINTOHEX}(\text{expression}) \]

Syntax Elements

- **expression**: The value to be converted to a hexadecimal value.

Description

Converts a binary value to a hexadecimal value of data type NVARCHAR. If the input value is not a binary value, then it is first converted to a binary value.

Example

The following example converts the binary value \textit{AB} to a hexadecimal NVARCHAR value \textit{4142}:

\[
\text{SELECT BINTOHEX('AB')} \text{ "bintohex" FROM DUMMY;}
\]

Related Information

- Expressions [page 56]
- Character String Data Types [page 36]
4.9.1.25 BINTONHEX Function (String)

Converts a binary value to a hexadecimal value as an NVARCHAR data type.

Syntax

BINTONHEX(<expression>)

Syntax Elements

expression The value to be converted to a hexadecimal value.

Description

Converts a binary value to a hexadecimal value as an NVARCHAR data type. The input value is converted to a binary value first if it is not a binary value.

Example

The following example converts the binary value AB to a hexadecimal NVARCHAR value 4142.

SELECT BINTONHEX('AB') "bintonhex" FROM DUMMY;

Related Information

Expressions [page 56]
Character String Data Types [page 36]
4.9.1.26 BINTOSTR Function (String)

Converts a VARBINARY string to a character string.

Syntax

```
BINTOSTR(<varbinary_string>)
```

Syntax Elements

- `varbinary_string` The VARBINARY string to be converted to a character string.

Description

Converts a VARBINARY string `<varbinary_string>` to a character string with CESU-8 encoding.

Example

This example converts the VARBINARY string `416E74` to a CESU-8 encoded character string, and returns the value `Ant`:

```
SELECT BINTOSTR ('416E74') "bintostr" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]
4.9.1.27 BITAND Function (Numeric)

Performs an AND operation on the bits of two arguments.

Syntax

BITAND(<value1>, <value2>)

Syntax Elements

- value1: The argument must be an INTEGER or a VARBINARY value.
- value2: The argument must be an INTEGER or a VARBINARY value.

Description

Performs an AND operation on the bits of the arguments <value1> and <value2>.

The BITAND function works a bit differently from the BITOR, BITXOR, and BITNOT functions. The BITAND function converts string arguments into BIGINT, whereas the other functions convert string arguments into INT.

The BITAND function returns a result along the argument’s type.

Example

The following example returns the value 123:

```
SELECT BITAND (255, 123) "bitand" FROM DUMMY;
```

Related Information

Expressions [page 56]
Numeric Data Types [page 48]
4.9.1.28 BITCOUNT Function (Numeric)

Counts the number of set bits of an expression.

Syntax

```
BITCOUNT(<expression>)
```

Syntax Elements

- **expression**: Specifies the expression that the function counts the number of set bits of. `<expression>` must be an INTEGER or a VARBINARY value.

Description

Counts the number of set bits of the argument `<expression>`.
The BITCOUNT function returns an INTEGER value.

Example

The following example counts the bits for 255, and returns the value 8:

```
SELECT BITCOUNT (255) "bitcount" FROM DUMMY;
```

Related Information

- Expressions [page 56]
- Numeric Data Types [page 48]
4.9.1.29 BITNOT Function (Numeric)

Performs a bitwise NOT operation on the bits of an expression.

Syntax

| BITNOT (<expression>) |

Syntax Elements

**expression** Specifies the expression that the function performs a bitwise NOT operation on. `<expression>` must be an INTEGER value.

Description

Performs a bitwise NOT operation on the bits of the argument `<expression>`.

The BITNOT functions works a bit differently than the BITAND function. The BITAND function converts string arguments into BIGINT, whereas the BITNOT function converts string arguments into INT.

The BITNOT function returns a result along the argument’s type.

Example

The following example performs a BITNOT operation on 255, and returns the value -256 for "bitnot":

```sql
SELECT BITNOT (255) "bitnot" FROM DUMMY;
```

Related Information

Expressions [page 56]
Numeric Data Types [page 48]
4.9.1.30 BITOR Function (Numeric)

This function performs an OR operation on the bits of two arguments.

Syntax

```
BITOR(<expression1>, <expression2>)
```

Syntax Elements

- **expression1**: An argument for the bitwise OR operation that must be an INTEGER or a VARBINARY value.
- **expression2**: An argument for the bitwise OR operation that must be an INTEGER or a VARBINARY value.

Description

This function performs an OR operation on the bits of the arguments `<expression1>` and `<expression2>`. The BITOR function works a bit differently than the BITAND function. The BITAND function converts string arguments into BIGINT, whereas the BITOR function converts string arguments into INT.

The BITOR function returns a result along the argument's type.

Example

The following example performs a bitwise OR operation for the arguments 255 and 123, and returns the value 255 for "bitor":

```
SELECT BITOR (255, 123) "bitor" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]
4.9.1.31 BITSET Function (Numeric)

Sets a specific number of bits to 1 in a target number from a specified 1-based index position.

Syntax

```
BITSET(<target_num>, <start_bit>, <num_to_set>)
```

Syntax Elements

- **target_num**
  The VARBINARY number where the bits are to be set.
  
  `<target_num> ::= <string_literal>`

- **start_bit**
  A 1-based index position where the first bit is to be set.
  
  `<start_bit> ::= <unsigned_integer>`

- **num_to_set**
  The number of bits to be set in the target number.
  
  `<num_to_set> ::= <unsigned_integer>`

Description

Sets `<num_to_set>` bits to 1 in `<target_num>` from the `<start_bit>` position.

Example

The following example returns the value E000 for "bitset":

```
SELECT BITSET ('0000', 1, 3) "bitset" FROM DUMMY;
```
Related Information

Numeric Data Types [page 48]

4.9.1.32  BITUNSET Function (Numeric)

Sets a specified number of bits to 0 in a target number from a specified 1-based index position.

Syntax

```
BITUNSET(<target_num>, <start_bit>, <num_to_unset>)
```

Syntax Elements

- **target_num**
  The VARBINARY number where the bits are to be unset.

  `<target_num> ::= <string_literal>`

- **start_bit**
  A 1-based index position where the first bit is to be unset.

  `<start_bit> ::= <unsigned_integer>`

- **num_to_set**
  The number of bits to be unset in the target number.

  `<num_to_set> ::= <unsigned_integer>`

Description

Sets `<num_to_unset>` bits to 0 in `<target_num>` from the `<start_bit>` position.
Example

The following example returns the value 1FFF for "bitunset":

```
SELECT BITUNSET ('ffff', 1, 3) "bitunset" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.33 BITXOR Function (Numeric)

Performs an XOR operation on the bits of two arguments.

Syntax

```
BITXOR(<expression1>, <expression2>)
```

Syntax Elements

- **expression1** An argument that must be an INTEGER or a VARBINARY value.
- **expression2** An argument for the bitwise OR operation that must be an INTEGER or a VARBINARY value.

Description

Performs an XOR operation on the bits of the arguments <expression1> and <expression2>.

The BITXOR functions works a bit differently than the BITAND function. The BITAND function converts string arguments into BIGINT, whereas the BITXOR function converts string arguments into INT.

The BITXOR function returns a result along the argument’s type.
Example

The following example performs a bitwise XOR operation for the arguments 255 and 123, and returns the value 132 for "bitxor":

```
SELECT BITXOR (255, 123) "bitxor" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.34 CARDINALITY Function (Array)

Returns the number of elements in a specified array.

Syntax

```
CARDINALITY(<array_value_expression>)
```

Syntax Elements

- `array_value_expression`: Specifies the array that the function returns the number of elements for.

Description

Returns the number of elements in `<array_value_expression>`.

Example

The following example returns the number of elements (3 and 4, respectively) contained in two arrays.

```sql
CREATE COLUMN TABLE ARRAY_TEST (IDX INT, VAL INT ARRAY);
INSERT INTO ARRAY_TEST VALUES (1, ARRAY(1, 2, 3));
INSERT INTO ARRAY_TEST VALUES (2, ARRAY(10, 20, 30, 40));
SELECT CARDINALITY(VAL) "cardinality" FROM ARRAY_TEST;
```
Related Information

Expressions [page 56]

4.9.1.35 CAST Function (Data Type Conversion)

Returns the value of an expression converted to a supplied data type.

Syntax

```
CAST( <expression> AS <data_type>[ ( <length> ) ] )
```

Syntax Elements

- **expression**
  Specifies the expression to be converted.

- **data_type**
  Specifies the target data type.

```
<data_type> ::= BOOLEAN | TINYINT | SMALLINT | INTEGER | BIGINT | DECIMAL | SMALLDECIMAL | REAL | DOUBLE | ALPHANUM | VARCHAR | NVARCHAR | DAYDATE | DATE | TIME | SECONDDATE | TIMESTAMP | LOB | BINARY | GEOMETRY
```

- **length** Specifies the maximum length of the string in characters. For example, `CAST (A AS VARCHAR(10))`. 

SAP HANA SQL Reference Guide for SAP HANA Platform
SQL Reference
PUBLIC 125
Description

Returns the value of an expression converted to a supplied data type.

Examples

The following example converts the value 7 to the VARCHAR value 7:

```
SELECT CAST (7 AS VARCHAR) "cast" FROM DUMMY;
```

The following example converts the value 10.5 to the INTEGER value 10, truncating the mantissa.

```
SELECT CAST (10.5 AS INTEGER) "cast" FROM DUMMY;
```

Bind a parameter as BIGINT. Without the CAST function, the data type of the parameter is ambiguous.

```
CREATE COLLECTION C1;
INSERT INTO C1 VALUES({A:1,B:2});
INSERT INTO C1 VALUES({A:'ABC'});
SELECT * FROM C1 WHERE A = CAST (?) AS BIGINT;
```

Related Information

Expressions [page 56]
Data Type Conversion [page 37]

4.9.1.36 CEIL Function (Numeric)

Returns the first integer that is greater than or equal to the specified value.

Syntax

```
CEIL(<number>)
```

Syntax Elements


\[\text{number}\] Specifies the number that the function returns the integer for.
Description

Returns the first integer that is greater than or equal to the value of \(<n>\).

Example

The following example returns the value 15 for "ceiling":

```
SELECT CEIL (14.5) "ceiling" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.37 CHAR Function (String)

Returns the character that has the ASCII value of the specified number.

Syntax

```
CHAR(<number>)
```

Syntax Elements

-\( \text{number} \): Specifies the number with the ASCII value that the function converts into a character.

Description

Returns the character that has the ASCII value of the specified number.
Example

This example converts three ASCII values into characters and concatenates the results, returning the string Ant:

```
SELECT CHAR (65) || CHAR (110) || CHAR (116) "character" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]

4.9.1.38 COALESCE Function (Miscellaneous)

Returns the first non-NULL expression from a specified list.

Syntax

```
COALESCE(<expression_list>)
```

Syntax Elements

- `expression_list` Specifies the expressions to return the first non-NULL expression from.

Description

Returns the first non-NULL expression from a list. At least two expressions must be contained in `<expression_list>`, and all expressions must be comparable. The result is NULL if all the expressions are NULL.

Example

```
CREATE ROW TABLE coalesce_example (ID INT PRIMARY KEY, A REAL, B REAL);
INSERT INTO coalesce_example VALUES(1, 100, 80);
INSERT INTO coalesce_example VALUES(2, NULL, 63);
```
The example above returns the results below:

<table>
<thead>
<tr>
<th>ID</th>
<th>A</th>
<th>B</th>
<th>coalesce</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100.0</td>
<td>80.0</td>
<td>100.0</td>
</tr>
<tr>
<td>2</td>
<td>NULL</td>
<td>63.0</td>
<td>69.30000305175781</td>
</tr>
<tr>
<td>3</td>
<td>NULL</td>
<td>NULL</td>
<td>50.0</td>
</tr>
</tbody>
</table>

**Related Information**

**Expressions [page 56]**

### 4.9.1.39 CONCAT Function (String)

Returns a combined string consisting of two specified strings.

**Syntax**

```
CONCAT(<string1>, <string2>)
```

**Syntax Elements**

- **string1** Specifies the first string to combine.
- **string2** Specifies the second string to combine.

**Description**

Returns a combined string consisting of `<string1>` followed by `<string2>`. The concatenation operator (||) is identical to this function.

The maximum length of the concatenated string is 8,388,607. If the string length is longer than the maximum length, an exception is thrown. Exceptionally, an implicit truncation is performed when converting an (N)CLOB typed value with a size greater than the maximum length of an (N)VARCHAR typed value.
If one or both arguments are NULL, the function returns NULL.

Example

This example concatenates the specified string arguments and returns the value Cat:

```sql
SELECT CONCAT ('C', 'at') "concat" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]

4.9.1.40 CONCAT_NAZ Function (String)

Returns a combined, non-null value string consisting of two specified strings.

Syntax

```
CONCAT_NAZ(<string1>, <string2>)
```

Syntax Elements

- `string1` Specifies the first string to combine.
- `string2` Specifies the second string to combine.

Description

Returns a combined value string consisting of `<string1>` followed by `<string2>`. If one value is NULL, the other value is returned. If both values are NULL, NULL is returned.

The maximum length of the concatenated string is 8,388,607. If the string length is longer than the maximum length, then an exception is thrown. Exceptionally, an implicit truncation is performed when converting an (N)CLOB typed value with a size greater than the maximum length of an (N)VARCHAR typed value.
Example

The following example returns the AB:

```
SELECT CONCAT_NAZ ('A', 'B') "concat" FROM DUMMY;
```

<table>
<thead>
<tr>
<th>concat</th>
<th>AB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following example returns C:

```
SELECT CONCAT_NAZ ('C', null) "concat" FROM DUMMY;
```

<table>
<thead>
<tr>
<th>concat</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following example returns the NULL value:

```
SELECT CONCAT_NAZ (null, null) "concat" FROM DUMMY;
```

<table>
<thead>
<tr>
<th>concat</th>
<th>NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Related Information

Character String Data Types [page 36]
Character String Data Types [page 36]

4.9.1.41 CONVERT_CURRENCY Function (Miscellaneous)

Calculates values in different currencies.

Syntax

```
CONVERT_CURRENCY( <named_parameter_value>[ [, <named_parameter_value>] ... ])
```

Syntax Elements

```
(named_parameter_value)
```
Specifies the parameters for the conversion.

```
<named_parameter_value> ::=   "<field_reference_parameter>" => <expression>    | "<const_string_parameter>" => <const_string>
```

You use field reference parameters to refer to table columns for the conversion. If the `<field_reference_parameter>` or `<const_string_parameter>` parameter contain an empty value, the conversion fails and the parameter remains unset. For example,

```
"TARGET_UNIT" => ''
```

results in TARGET_UNIT remaining unset and the execution fails.

<table>
<thead>
<tr>
<th>Field Reference Parameter</th>
<th>Description</th>
<th>Mandatory</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMOUNT</td>
<td>The column identifier containing the values to be converted.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>SOURCE_UNIT</td>
<td>Describes the input unit. A constant string is also accepted using single quotations.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>TARGET_UNIT</td>
<td>Defines a three character string that is used to separate tenants within ERP system tables. This is used in the conversion tables to select the correct rows for each user. A column identifier is also accepted using double quotations. This parameter is mandatory, as the CLIENT session context variable is not used by this command.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>CLIENT</td>
<td>Defines a three character string that is used to separate tenants within ERP system tables. This is used in the conversion tables to select the correct rows for each user. A column identifier is also accepted using double quotations. This parameter is mandatory, as the CLIENT session context variable is not used by this command.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>
### Field Reference Parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Mandatory</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TARGET_UNIT</td>
<td>The column identifier describing the target unit. A constant string is also accepted using single quotations.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>REFERENCE_DATE</td>
<td>The column identifier describing the currency reference date. A constant string is also accepted using single quotations.</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>

### Constant Parameter Name

<const_string_parameter> defines constant parameter values used in the conversion.

<table>
<thead>
<tr>
<th>Constant Parameter Name</th>
<th>Description</th>
<th>Mandatory</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA</td>
<td>Defines the schema that contains the conversion tables used for the conversion.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>DATABASE</td>
<td>Defines the tenant where the conversion tables are located (for example the TCUR* tables of the ERP currency conversion). If not specified, the conversion uses the current database.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>METHOD</td>
<td>Possible values: ERP, Banking</td>
<td>No</td>
<td>ERP</td>
</tr>
<tr>
<td>BID_ASK_TYPE</td>
<td>Possible values: bid, ask, mid</td>
<td>No</td>
<td>(empty string)</td>
</tr>
<tr>
<td>Constant Parameter Name</td>
<td>Description</td>
<td>Mandatory</td>
<td>Default Value</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>MARKET_DATA_AREA</td>
<td>Defines the market data area as stored in the tables. This is mandatory for banking currency conversion.</td>
<td>No</td>
<td>(empty_string)</td>
</tr>
<tr>
<td>SYSTEM_TIME</td>
<td>Defines the system timestamp for time travel functionality.</td>
<td>No</td>
<td>Current system timestamp in GMT</td>
</tr>
<tr>
<td>CONVERSION_TYPE</td>
<td>Defines the conversion type as stored in the conversion tables. The conversion types available in your system vary according to the setup of your ERP system. In general, these are either M or EURX. Contact your system administrator for the details of your specific table configuration.</td>
<td>No</td>
<td>M</td>
</tr>
<tr>
<td>LOOKUP</td>
<td>The type of lookup to be performed. Possible values:</td>
<td>No</td>
<td>Regular</td>
</tr>
<tr>
<td></td>
<td>Regular</td>
<td></td>
<td>A regular conversion is performed.</td>
</tr>
<tr>
<td></td>
<td>Reverse</td>
<td></td>
<td>Performs a reverse conversion with the input units swapped.</td>
</tr>
</tbody>
</table>

SAP HANA SQL Reference Guide for SAP HANA Platform

SQL Reference

134  PUBLIC
<table>
<thead>
<tr>
<th>Constant Parameter Name</th>
<th>Description</th>
<th>Mandatory</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| ERROR_HANDLING         | Defines how the system handles a situation where a row could not be converted. Possible values:  
  - fail on error: The conversion fails with an error.  
  - set to null: The output from the row that caused the error is set to NULL.  
  - keep unconverted: The input value is returned.  

Error handling covers cases where all mandatory parameters are provided. A conversion with empty or missing mandatory parameters fails independent of the error handling. |

| ACCURACY | Defines the rounding behavior of the system. Possible values:  
  - compatibility: Mimics ERP behavior by rounding the result.  
  - highest: Keeps as many digits as possible in the result. | No | Compatibility |


<table>
<thead>
<tr>
<th>Constant Parameter Name</th>
<th>Description</th>
<th>Mandatory</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE_FORMAT</td>
<td>Defines the format in which the reference date is presented. Possible values</td>
<td>No</td>
<td>Auto detect</td>
</tr>
<tr>
<td></td>
<td>auto detect</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Attempt automatic detection of the date format.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>normal</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Date is provided in a regular format</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>inverted</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Date is provided in inverted SAP legacy format.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constant Parameter Name</td>
<td>Description</td>
<td>Mandatory</td>
<td>Default Value</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>---------------------</td>
</tr>
<tr>
<td>STEPS</td>
<td>Defines the steps that should be included in the conversion process. You provide a comma delimited list of the steps to be included, and the order of the steps is irrelevant.</td>
<td>No</td>
<td>shift, convert</td>
</tr>
<tr>
<td></td>
<td><strong>shift</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enables a decimal shift according to the source currency selected. For example, if the source currency has 0 valid digits according to PRECISIONS_TABLE, each value needs to be multiplied by 100 because in SAP ERP systems, values are stored using two digits. This has to be done to convert ERP values to their correct numerical representation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>convert</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Triggers the actual conversion from the source to the target currency.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>round</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rounds the converted value to the number of digits of the target currency. Use this step carefully if subsequent aggregations take place on the number, as rounding errors could accumulate.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>shift_back</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>While shift changes the decimals from two to the configured precision of the source currency, shift_back</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constant Parameter Name</td>
<td>Description</td>
<td>Mandatory</td>
<td>Default Value</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>changes them back to two, but from the target currency. If error handling is set to keep unconverted the output currency might be the source instead of the target currency. In case of an error, the rounding and the shift_back are done with respect to the source currency and the conversion is dropped. This renders all steps redundant, and yields the input value again.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONFIGURATION_TABLE</td>
<td>The table identifier of the conversion type configuration. If the table resides in a different tenant, provide a fully qualified name for the table or use the DATABASE parameter.</td>
<td>No</td>
<td>TCURV</td>
</tr>
<tr>
<td>PRECISIONS_TABLE</td>
<td>The table identifier of the precision table. If the table resides in a different tenant, provide a fully qualified name for the table or use the DATABASE parameter.</td>
<td>No</td>
<td>TCURX</td>
</tr>
<tr>
<td>NOTATIONS_TABLE</td>
<td>The table identifier of the table that stores notations. If the table resides in a different tenant, provide a fully qualified name for the table or use the DATABASE parameter.</td>
<td>No</td>
<td>TCURN</td>
</tr>
</tbody>
</table>

Additional Parameters and Defaults for Method ERP
<table>
<thead>
<tr>
<th>Constant Parameter Name</th>
<th>Description</th>
<th>Mandatory</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIGURATION_TABLE</td>
<td>The table identifier of the conversion type configuration. If the table resides in a different tenant, provide a fully qualified name for the table or use the DATABASE parameter.</td>
<td>No TCURV</td>
<td>TCURV</td>
</tr>
<tr>
<td>RATES_TABLE</td>
<td>The table identifier of the conversion rates table. If the table resides in a different tenant, provide a fully qualified name for the table or use the DATABASE parameter.</td>
<td>No TCURR</td>
<td>TCURR</td>
</tr>
<tr>
<td>PREFACTORS_TABLE</td>
<td>The table identifier of the pre-factors table. If the table resides in a different tenant, provide a fully qualified name for the table or use the DATABASE parameter.</td>
<td>No TCURF</td>
<td>TCURF</td>
</tr>
</tbody>
</table>

**Additional Parameters and Defaults for Method Banking**

<table>
<thead>
<tr>
<th>Constant Parameter Name</th>
<th>Description</th>
<th>Mandatory</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARKET_DATA_AREA</td>
<td>Defines the market data area as stored in the tables. This is mandatory for banking currency conversion.</td>
<td>Yes</td>
<td>(empty_string)</td>
</tr>
<tr>
<td>SYSTEM_TIME</td>
<td>Defines the system timestamp for time travel functionality. T</td>
<td>No</td>
<td>Current system timestamp in GM</td>
</tr>
<tr>
<td>CONFIGURATION_TABLE</td>
<td>The table identifier of the conversion type configuration. If the table resides in a different tenant, provide a fully qualified name for the table or use the DATABASE parameter.</td>
<td>No</td>
<td>/BA1/TF4_FXRTTYP</td>
</tr>
<tr>
<td>RATES_TABLE</td>
<td>The table identifier of the conversion rates table. If the table resides in a different tenant, provide a fully qualified name for the table or use the DATABASE parameter.</td>
<td>No</td>
<td>/BA1/F4_FXRATES</td>
</tr>
</tbody>
</table>
**Constant Parameter Name** | **Description** | **Mandatory** | **Default Value**
---|---|---|---
PREFACTORS_TABLE | The table identifier of the pre-factors table. If the table resides in a different tenant, provide a fully qualified name for the table or use the DATABASE parameter. | No | /BA1/TF4_FXCVFCT

**Description**

The CONVERT_CURRENCY function provides an efficient method to calculate values in different currencies. The CONVERT_CURRENCY function is an SQL representation of the SQLScript built-in function CE_CONVERSION, and internally uses CE_CONVERSION for computation.

The HANA currency conversion provides two different methods. The ERP method, which stores the cross rates in table TCURR with 9 digits, and the Banking method, which stores the rates in table /BA1/F4_FXRATES with 15 digits. In both cases, you can alternatively use tables that store the cross rates in a data type with more digits, such as DECIMAL(25,21).

To use the CONVERT_CURRENCY function, the currency conversion tables TCURV, TCURX, TCURN, TCURR, and TCURF must be available in the SAP HANA database for method ERP. Alternatively, the currency conversion tables /BA1/TF4_FXRTTYP, TCURX, TCURN, /BA1/F4_FXRATES, and /BA1/TF4_FXCVFCT must be available for method Banking. If you have stored the rates information in tables with a different layout, you can use views to map the table content to the format used in the ERP or Banking tables.

The CONVERT_CURRENCY function does not support some conversion functions. It only supports columns XINVR, BWAER, XBWRL and XEURO from customizing table TCURV. An error message appears when one of the unsupported fields BKUZU, GKUZU or XBWRL are maintained in table TCURV. Refer to SAP Note 2792149 - Currency/unit conversion error: conversion type '<conversion type>' has unsupported 'BKUZU' or 'GKUZU' type set.

**Example**

Create a table and populate it with two example currency amounts.

```sql
CREATE ROW TABLE sample_input (price DECIMAL(15,2),
source_unit VARCHAR(4),
target_unit VARCHAR(4),
ref_date VARCHAR(10));
INSERT INTO sample_input VALUES (1.0, 'SRC', 'TRG', '2011-01-01');
INSERT INTO sample_input VALUES (1.0, 'SRC', 'TRG', '2011-02-01');
```

Convert the values in the currency table using conversion tables contained in the SYSTEM schema in tenant NDB.

```sql
SELECT CONVERT_CURRENCY(amount=>price,
```

SAP HANA SQL Reference Guide for SAP HANA Platform
Convert the values by using the Banking method.

```sql
SELECT *, CONVERT_CURRENCY(method=>'Banking', -- new
market_data_area=>'S000', -- new (and mandatory for Banking)
amount=>ext_limit,
source_unit =>ext_limit_curr,
schema => 'SAPCOB',
target_unit=> 'EUR',
reference_date => business_day,
error_handling=> 'set to null',
client => '150') as converted from v_fx_input
```

Convert the values by using the Banking method and system_time.

```sql
SELECT *, CONVERT_CURRENCY(method=>'Banking', -- new
market_data_area=>'S000', -- mandatory (for Banking)
bid_ask_type=>'MID', -- optional
system_time=>to_timestamp('2011-05-11 12:59.999','YYYY-MM-DD HH:SS.FF3') -- optional
amount=>ext_limit,
source_unit =>ext_limit_curr,
schema => 'SAPCOB',
target_unit=> 'EUR',
reference_date => business_day,
error_handling=> 'set to null',
client => '150') as converted from v_fx_input
```

Related Information

- SAP Note 2792149
- Expressions [page 56]
- Currency Translation
- Associate Measures with Currency
4.9.1.42 CONVERT_UNIT Function (Miscellaneous)

Converts the specified source units to specified target units.

Syntax

```sql
CONVERT_UNIT( <named_parameter_value>, ... )
```

Syntax Elements

`named_parameter_value`

Named value of field reference, or constant string, parameters.

```sql
<named_parameter_value> ::= "<field_reference_parameter>" => <expression> | "<const_string_parameter>" => <const_string>
```

`<field_reference_parameter>` ::= QUANTITY | SOURCE_UNIT | TARGET_UNIT | CLIENT

`<const_string_parameter>` ::= SCHEMA | DATABASE | ERROR_HANDLING | RATES_TABLE | DIMENSION_TABLE

Field Reference Parameter

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Mandatory</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUANTITY</td>
<td>The column to be converted.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>SOURCE_UNIT</td>
<td>The column identifier describing the input unit. A constant string is also accepted using single quotations.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>TARGET_UNIT</td>
<td>The column identifier describing the target unit. A constant string is also accepted using single quotations.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Mandatory</td>
<td>Default Value</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>CLIENT</td>
<td>Defines a three character string that is used to separate tenants within ERP system tables. This is used in the conversion tables to select the correct rows for each user. A column identifier is also accepted using double quotations. This parameter is mandatory, as the CLIENT session context variable is not used by this command.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>Defines the schema that contains the conversion tables used for the conversion.</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>DATABASE</td>
<td>Defines the tenant where the conversion tables are located. If not specified, the conversion uses the current database.</td>
<td>No</td>
<td>None</td>
</tr>
<tr>
<td>ERROR_HANDLING</td>
<td>Defines how the system handles a situation where a row cannot be converted.</td>
<td>No</td>
<td>fail on error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>fail on error</td>
<td>The conversion fails with an error.</td>
</tr>
<tr>
<td>set to null</td>
<td>The output from the row that caused the error is set to NULL.</td>
</tr>
<tr>
<td>keep unconverted</td>
<td>The input value is returned.</td>
</tr>
</tbody>
</table>
### Description

The `CONVERT_UNIT` function is an SQL representation of the SQL Script built-in function `CE_CONVERSION`, and internally uses `CE_CONVERSION` for computation.

To use the `CONVERT_UNIT` function, the unit conversion tables `T006` and `T006D` must be available in the SAP HANA database. For other SAP HANA databases, replicate the `T006` and `T006D` tables from an SAP ERP system.

### Example

```sql
CREATE ROW TABLE sample_input (  quant DECIMAL(15,2),  source_unit VARCHAR(4),  target_unit VARCHAR(4) ) ;  INSERT INTO sample_input VALUES (1.0, 'SRC', 'TRG');  INSERT INTO sample_input VALUES (1.0, 'SRC', 'TRG');  SELECT CONVERT_UNIT("QUANTITY"=>quant ,  "SOURCE_UNIT" =>source_unit  ,  "SCHEMA" => 'SYSTEM'  ,  "DATABASE" => 'NDB'  ,  "TARGET_UNIT" => target_unit  ,  "ERROR_HANDLING"=>'set to null'  ,  "CLIENT" => '000') AS converted FROM sample_input;```
4.9.1.43 CORR Function (Aggregate)

Computes the Pearson product momentum correlation coefficient between two columns. This function can also be used as a window function.

Syntax

Aggregate function:

CORR(<column1>, <column2>)

Window function:

CORR(<column1>, <column2>) <window_specification>

Syntax Elements

- column1 and column2
  Specifies the columns providing the input data for the correlation.
  The values of <column1> and <column2> can be of any numeric type.
- window_specification
  Defines a window on the data over which the function operates. For <window_specification>, see Window Functions and the Window Specification [page 424].

Description

Computes the Pearson product momentum correlation coefficient between two columns.

The result ranges from -1 to 1, depending on the correlation, or NULL if a correlation cannot be computed.

The result can return NULL for one of the following reasons:

- Less than two value pairs are correlated after NULLs have been removed
- There is zero variance in at least one of the two columns
**Examples**

The examples below assume that a correlation table has been created with the following values:

```sql
CREATE COLUMN TABLE correlationTable (     ts_id VARCHAR(20),
    DATE DAYDATE,
    value1 DOUBLE,
    value2 DOUBLE);
INSERT INTO correlationTable VALUES ('A', '2014-10-01', 1, 1);
INSERT INTO correlationTable VALUES ('A', '2014-10-02', 2, 2);
INSERT INTO correlationTable VALUES ('A', '2014-10-04', 3, 3);
INSERT INTO correlationTable VALUES ('B', '2014-10-07', 1, 3);
INSERT INTO correlationTable VALUES ('B', '2014-10-11', 2, 2);
INSERT INTO correlationTable VALUES ('B', '2014-10-21', 3, 1);
```

1. The following aggregate function example returns the correlation between the `ts_id` column and the columns `value1` and `value2`.

   ```sql
   SELECT ts_id, CORR(value1, value2) FROM correlationTable GROUP BY ts_id;
   ```

   The results are as follows:

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR(VALUE1, VALUE2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
</tbody>
</table>

2. The following WHERE clause example returns the correlation between the `ts_id` column and the columns `value1` and `value2` only for rows where `ts_id` equals `A`.

   ```sql
   SELECT ts_id, CORR(value1, value2) FROM correlationTable WHERE ts_id='A' GROUP BY ts_id;
   ```

   The results are as follows:

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR(VALUE1, VALUE2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
</tbody>
</table>

3. The example below uses a window function.

   ```sql
   SELECT ts_id, CORR(value1, value2) OVER (PARTITION BY ts_id) FROM correlationTable;
   ```

   The results are as follows:

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR(VALUE1, VALUE2) OVER(PARTITION BY TS_ID)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
</tbody>
</table>
4. The example below uses a sliding window function.

```
SELECT ts_id, CORR(value1, value2) OVER (PARTITION BY ts_id ORDER BY date)
FROM correlationTable ORDER BY ts_id;
```

The results are as follows:

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR(VALUE1, VALUE2)OVER(PARTITIONBYTS_IDORDERBYDATE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>?</td>
</tr>
<tr>
<td>A</td>
<td>0.9999999999999998</td>
</tr>
<tr>
<td>A</td>
<td>0.9999999999999998</td>
</tr>
<tr>
<td>B</td>
<td>?</td>
</tr>
<tr>
<td>B</td>
<td>-0.9999999999999998</td>
</tr>
<tr>
<td>B</td>
<td>-0.9999999999999998</td>
</tr>
</tbody>
</table>

5. The example below uses a ROWS BETWEEN clause.

```
SELECT ts_id, CORR(value1, value2) OVER (PARTITION BY ts_id ORDER BY date
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) from correlationTable;
```

The results are as follows:

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR(VALUE1, VALUE2)OVER(PARTITIONBYTS_IDORDERBYDATEROWS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>?</td>
</tr>
<tr>
<td>A</td>
<td>0.9999999999999998</td>
</tr>
<tr>
<td>A</td>
<td>0.9999999999999998</td>
</tr>
<tr>
<td>B</td>
<td>?</td>
</tr>
<tr>
<td>B</td>
<td>-0.9999999999999998</td>
</tr>
<tr>
<td>B</td>
<td>-0.9999999999999998</td>
</tr>
</tbody>
</table>

**Related Information**

- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- Aggregate Functions [page 414]
- AUTO_CORR Function (Aggregate) [page 108]
4.9.1.44 **CORR_SPEARMAN Function (Aggregate)**

Returns the Spearman's rank correlation coefficient of the values found in the corresponding rows of two columns. This function can also be used as a window function.

### Syntax

**Aggregate function:**

```
CORR_SPEARMAN(<column1>, <column2>)
```

**Window function:**

```
CORR_SPEARMAN(<column1>, <column2>) <window_specification>
```

### Syntax Elements

- **column1 and column2**
  - Specifies the columns providing the input data for the correlation.
  - The values of `<column1>` and `<column2>` can contain number or character types.

- **window_specification**
  - Defines a window on the data over which the function operates. For `<window_specification>`, see *Window Functions and the Window Specification* [page 424].

### Description

Returns the Spearman's rank correlation coefficient of the values found in the corresponding rows of `<column1>` and `<column2>.

Although this function is grouped with the aggregate functions, its optional OVER clause positions it as a windows function as well.

The result ranges from -1 to 1, depending on the correlation, or NULL if a correlation could not be computed.

The result can return NULL for one of the following reasons:

- Less than two value pairs are correlated after NULLs have been removed
• There is zero variance in at least one of the two columns
Whenever a NULL value is found then both the NULL value and the corresponding value of the other input column are ignored.

**Examples**

The example below returns -1.

```sql
CREATE COLUMN TABLE A (date DAYDATE, val INT);
INSERT INTO A VALUES ('2014-10-01', 100);
INSERT INTO A VALUES ('2014-10-02', 200);
INSERT INTO A VALUES ('2014-10-03', 300);
CREATE COLUMN TABLE B (date DAYDATE, val INT);
INSERT INTO B VALUES ('2014-10-01', 300);
INSERT INTO B VALUES ('2014-10-02', 200);
INSERT INTO B VALUES ('2014-10-03', 100);
SELECT CORR_SPEARMAN(A.val, B.val) "corr" FROM A, B WHERE A.date = B.date;
```

The examples below assume that the correlation table has been created with the following values:

```sql
CREATE COLUMN TABLE correlationSpearmanTable (     ts_id VARCHAR(20),
  date DAYDATE,
  value1 DOUBLE,
  value2 DOUBLE);
INSERT INTO correlationSpearmanTable VALUES ('A', '2014-10-01', 34.345, 45.345);
INSERT INTO correlationSpearmanTable VALUES ('A', '2014-10-02', 27.145, 28.893);
INSERT INTO correlationSpearmanTable VALUES ('A', '2014-10-02', 48.312, 28.865);
INSERT INTO correlationSpearmanTable VALUES ('A', '2014-10-03', 94.213, 58.854);
INSERT INTO correlationSpearmanTable VALUES ('A', '2014-10-03', 16.567, 28.231);
INSERT INTO correlationSpearmanTable VALUES ('A', '2014-10-03', 38.894, 94.378);
INSERT INTO correlationSpearmanTable VALUES ('B', '2014-10-04', 45.643, 76.987);
INSERT INTO correlationSpearmanTable VALUES ('B', '2014-10-04', 53.345, 50.893);
INSERT INTO correlationSpearmanTable VALUES ('B', '2014-10-04', 66.342, 48.342);
INSERT INTO correlationSpearmanTable VALUES ('B', '2014-10-04', 76.432, 37.234);
INSERT INTO correlationSpearmanTable VALUES ('B', '2014-10-05', 88.432, 23.242);
INSERT INTO correlationSpearmanTable VALUES ('B', '2014-10-05', 93.234, 13.132);
```

1. Execute the aggregate function example below:

```sql
SELECT ts_id, CORR_spearman(value1, value2) FROM correlationSpearmanTable
GROUP BY ts_id;
```

The results are as follows:

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR_SPEARMAN(VALUE1,VALUE2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.542857</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
</tbody>
</table>

2. Execute the window function below:

```sql
SELECT ts_id, CORR_spearman(value1, value2) OVER (PARTITION BY ts_id) FROM correlationSpearmanTable;
```
The results are as follows:

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR_SPEARMAN(VALUE1,VALUE2)OVER(PARTITION-BYTS_ID)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.542857</td>
</tr>
<tr>
<td>A</td>
<td>0.542857</td>
</tr>
<tr>
<td>A</td>
<td>0.542857</td>
</tr>
<tr>
<td>A</td>
<td>0.542857</td>
</tr>
<tr>
<td>A</td>
<td>0.542857</td>
</tr>
<tr>
<td>A</td>
<td>0.542857</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
</tbody>
</table>

3. Execute the sliding window example below:

```sql
SELECT ts_id, CORR_spearman(value1, value2) OVER (PARTITION BY ts_id ORDER BY date) FROM correlationSpearmanTable ORDER BY ts_id;
```

The results are as follows:

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR_SPEARMAN(VALUE1,VALUE2)OVER(PARTITION-BYTS_IDORDERBYDATE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>?</td>
</tr>
<tr>
<td>A</td>
<td>-0.5</td>
</tr>
<tr>
<td>A</td>
<td>-0.5</td>
</tr>
<tr>
<td>A</td>
<td>0.542857</td>
</tr>
<tr>
<td>A</td>
<td>0.542857</td>
</tr>
<tr>
<td>A</td>
<td>0.542857</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
</tbody>
</table>
4. Execute the ROWS BETWEEN example below:

```sql
SELECT ts_id, CORR_spearman(value1, value2) OVER (PARTITION BY ts_id ORDER BY
date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) from
correlationSpearmanTable;
```

The results are as follows:

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR_SPEARMAN(VALUE1,VALUE2)OVER(PARTITION-BYTS_IDORDERBYDATE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR_SPEARMAN(VALUE1,VALUE2)OVER(PARTITION-BYTS_IDORDERBYDATEROWS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>?</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>-0.5</td>
</tr>
<tr>
<td>A</td>
<td>0.4</td>
</tr>
<tr>
<td>A</td>
<td>0.7</td>
</tr>
<tr>
<td>A</td>
<td>0.5428571428571428</td>
</tr>
<tr>
<td>B</td>
<td>?</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
</tbody>
</table>
5. Execute the group example below:

```sql
SELECT ts_id, CORR_spearman(value1, value2) OVER (PARTITION BY ts_id
ORDER BY date GROUPS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) FROM
correlationSpearmanTable;
```

The results are as follows:

<table>
<thead>
<tr>
<th>TS_ID</th>
<th>CORR_SPEARMAN(VALUE1,VALUE2)OVER(PARTITION-BYTS_IDORDERBYDATEGROUPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>?</td>
</tr>
<tr>
<td>A</td>
<td>-0.5</td>
</tr>
<tr>
<td>A</td>
<td>-0.5</td>
</tr>
<tr>
<td>A</td>
<td>0.5428571428571428</td>
</tr>
<tr>
<td>A</td>
<td>0.5428571428571428</td>
</tr>
<tr>
<td>A</td>
<td>0.5428571428571428</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>-1</td>
</tr>
</tbody>
</table>

**Related Information**

- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- Aggregate Functions [page 414]
- AUTO_CORR Function (Aggregate) [page 108]
- CORR Function (Aggregate) [page 145]
- CROSS_CORR Function (Aggregate) [page 157]
4.9.1.45  COS Function (Numeric)

Returns the cosine of the angle, in radians, for the specified argument.

Syntax

\[ \text{COS(<number>)} \]

Description

Returns the cosine of the angle \(<number>\), in radians.

Example

The following example returns the value 1.0 for \(\text{cos}\):

\[ \text{SELECT COS (0.0) "cos" FROM DUMMY;} \]

Related Information

Numeric Data Types [page 48]

4.9.1.46  COSH Function (Numeric)

Computes the hyperbolic cosine of the specified argument.

Syntax

\[ \text{COSH(<number>)} \]
Description

Computes the hyperbolic cosine of the numeric argument \(<number>\).

Example

The following example returns the value 1.1276259652063807 for "cosh":

```
SELECT COSH (0.5) "cosh" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.47 COT Function (Numeric)

Computes the cotangent of a specified number.

Syntax

```
COT(<number>)
```

Description

Computes the cotangent of a number \(<number>\), where \(<number>\) is an angle expressed in radians.

Example

The following example returns the value -0.8950829176379128 for "cot":

```
SELECT COT (40) "cot" FROM DUMMY;
```
4.9.1.48 COUNT Function (Aggregate)

Counts the number of rows returned by a query. This function can also be used as a window function.

Syntax

Aggregate function:

| COUNT (*) | COUNT ( [ ALL ] <expression> ) | COUNT ( DISTINCT <expression_list> ) |

Window function:

| COUNT (*) [ <window_specification> ] | COUNT ( [ ALL ] <expression> ) [ <window_specification> ] | COUNT ( DISTINCT <expression> ) [ <window_specification> ] |

Syntax Elements

*  
Returns the number of rows returned by the query, regardless of the value of those rows, and including duplicate values.

ALL  
Optional keyword reflecting the default behavior. That is, `COUNT(ALL <expression>)` and `COUNT(<expression>)` return the same result.

expression  
Specifies the input data for the function. The function returns the number of non-NULL values for the specified expression returned by the query, including duplicate values (unless DISTINCT is also specified).

eXpression_list  
Specifies the input data for the function as a comma-separated list of `<expression>`. `<expression_list>` is only supported for use with the DISTINCT clause.

| <expression_list>::= <expression> [, <expression> [,... ] ] |

DISTINCT  
Returns the number of distinct values returned by the query, excluding rows with all NULL values for that expression.
**window_specification** Defines a window on the data over which the function operates. For syntax, see Window Functions and the Window Specification [page 424].

**Description**

Result type based on input

<table>
<thead>
<tr>
<th>TINYINT</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>BIGINT</th>
<th>DECIMAL (p, s)</th>
<th>DECIMAL</th>
<th>REAL</th>
<th>DOUBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
<td>BIGINT</td>
<td>BIGINT</td>
<td>BIGINT</td>
<td>BIGINT</td>
<td>BIGINT</td>
<td>BIGINT</td>
</tr>
</tbody>
</table>

**Example**

The following statements create a table with data for example purposes:

```sql
DROP TABLE "MyProducts";
CREATE COLUMN TABLE "MyProducts"(
    "Product_ID" VARCHAR(10),
    "Product_Name" VARCHAR(100),
    "Category" VARCHAR(100),
    "Quantity" INTEGER,
    "Price" DECIMAL(10,2),
    PRIMARY KEY ("Product_ID")
);
INSERT INTO "MyProducts" VALUES('P1','Shirts', 'Clothes', 32, 20.99);
INSERT INTO "MyProducts" VALUES('P2','Jackets', 'Clothes', 16, 99.49);
INSERT INTO "MyProducts" VALUES('P3','Trousers', 'Clothes', 30, 32.99);
INSERT INTO "MyProducts" VALUES('P4','Coats', 'Clothes', 5, 129.99);
INSERT INTO "MyProducts" VALUES('P5','Purse', 'Accessories', 3, 89.49);
```

The following example returns 5, the number of rows in the MyProducts table:

```sql
SELECT COUNT(*) FROM "MyProducts";
```

The following example returns 2, the number of rows with distinct values in the Category column:

```sql
SELECT COUNT(DISTINCT "Category") FROM "MyProducts";
```

**Related Information**

- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- Aggregate Functions [page 414]
4.9.1.49 CROSS_CORR Function (Aggregate)

Computes all cross-correlation coefficients between two given expressions.

Syntax

```
CROSS_CORR(<expression1>, <expression2>, <maxLag>       { <series_orderby> | <order_by_clause> } ).{ POSITIVE_LAGS | NEGATIVE_LAGS | ZERO_LAG }
```

Syntax Elements

**expression1 and expression2**

Numeric values between which the cross-correlation is calculated.

**maxLag**

The `<maxLag>` parameter must be a positive integer that defines the number of cross-correlation coefficients to be returned.

```
<maxLag> ::= INTEGER
```

**series_orderby**

The SERIES clause can only be used with an equidistant series. For more information about the SERIES clause, see the CREATE TABLE statement and the SERIES_GENERATE function.

```
<series_orderby> ::= SERIES (<series_period> <series_equidistant_definition>)
```

**order_by_clause**

Specifies the sort order of the input rows.

```
<order_by_clause> ::= ORDER BY <order_by_expression> [, <order_by_expression> [...]]
<order_by_expression> ::= <column_name> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [ <collate_clause> ]
| <column_position> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [ <collate_clause> ]
| <collate_clause> ::= COLLATE <collation_name>
```

<collate_clause> specifies the collation to use for ordering values in the results. <collate_clause> can only be used on columns defined as NVARCHAR or VARCHAR. <collation_name> is one of the supported collation names listed in the COLLATIONS system view.
Description

Computes all cross-correlation coefficients between two given expressions.
The result is an array of cross-correlation coefficients of length `<maxLag>`.
If POSITIVE_LAGS is specified, then the cross-correlation coefficients with lags 1 `<maxLag>` are returned.
If NEGATIVE_LAGS is specified, then the cross-correlation coefficients with lags -1 `<maxLag>` are returned.
If ZERO_LAG is specified, a single value associated with lag 0 is returned.
This function output can be non-deterministic among tie values.

Example

Example 1 - Cross correlation

Execute the cross correlation example below:

```sql
CREATE COLUMN TABLE table1 ( ts_id INTEGER, number1 DOUBLE, number2 DOUBLE );
INSERT INTO table1 VALUES ('1', 1, 2);
INSERT INTO table1 VALUES ('2', 2, 1);
INSERT INTO table1 VALUES ('3', 1, 2);
SELECT CROSS_CORR(number1, number2, 10 ORDER BY ts_id) FROM table1;
```

The results are as follows:

```
CROSS_CORR(NUMBER1,NUMBER2,10ORDERBYTS_ID)
1.0, -1.0, 1.0
```

Example 2 - Cross correlation using a series descriptor

Execute the example below:

```sql
CREATE COLUMN TABLE TSeries( key INTEGER, ts TIMESTAMP, val1 DOUBLE, val2 DOUBLE, PRIMARY KEY(key, ts) )
    SERIES( SERIES KEY (key) EQUIDISTANT INCREMENT BY INTERVAL 1 DAY PERIOD FOR SERIES(ts) );
INSERT INTO TSeries VALUES (1, '2014-1-1', 1, 3);
INSERT INTO TSeries VALUES (2, '2014-1-3', 2, 4);
INSERT INTO TSeries VALUES (3, '2014-1-4', 4, 2);
INSERT INTO TSeries VALUES (4, '2014-1-5', 3, 1);
SELECT CROSS_CORR(val1, val2, 10 ORDER BY ts) FROM TSeries;
```

The results are as follows:

```
CROSS_CORR(VA1,VA2,10ORDERBYTS)
-1.0, -0.928571, -0.6, 0.5, -1.0
```
4.9.1.50 CUBIC_SPLINE_APPROX Function (Window)

Replaces null values by interpolating the gaps based on calculated cubic splines and linearly extrapolating any leading or trailing null values.

Syntax

  OVER ( { SERIES TABLE <series_table> [ PARTITION BY <col1> [,...] [ <window_order_by_clause> ]
         SERIES(...) [ PARTITION BY <col1> [,...] [ <window_order_by_clause> ]
         [ PARTITION BY <col1>[,...] <window_order_by_clause> ]
       ] } ] )

Syntax Elements

equation

Specifies the input data expression

series_table

Defines the series table.

<series_table> ::= [<schema_name>.]<table_name>
<schema_name> ::= <unicode_name>
<table_name> ::= <identifier>

<table_schema> must be a base table name, and not a correlation name.

BoundaryConditionArgument
Specifies the boundary conditions that are used for the cubic spline calculation. The default value is SPLINE_TYPE_NATURAL. If this parameter is omitted, then natural boundary conditions are used to calculate the interpolating cubic splines.

<BoundaryConditionArgument> ::= SPLINE_TYPE_NATURAL | SPLINE_TYPE_NOT_A_KNOT

ExtrapolationModeArgument

Defines the applied extrapolation mode. The default value is EXTRAPOLATION_NONE, which indicates that extrapolation does not occur if this parameter is omitted.

<ExtrapolationModeArgument> ::= EXTRAPOLATION_NONE | EXTRAPOLATION_LINEAR | EXTRAPOLATION_CONSTANT

The definition, default value, and use of ExtrapolationModeArgument in conjunction with the Value1Argument and Value2Argument is identical to the LINEAR_APPROX function except from the calculation of the slope.

When using EXTRAPOLATION_LINEAR mode, the slope of the extrapolation line is set equal to the slope of the calculated cubic splines at the boundary points. For example, the slope of the calculated spline at the first non-null value is taken as the slope of the extrapolating line for leading nulls. Similarly, the slope of the calculated spline at the last non-null value is taken as the slope of the extrapolating line for trailing nulls.

Cubic spline interpolation can be applied to all number types that can be cast into long double type, such as INT, LONG, DOUBLE, or FLOAT.

window_order_by_clause

Determines the sort order. This expression must not contain any NULL values or duplicates. See the syntax of this clause located in the Window Functions topic.

Description

Replaces null values by interpolating the gaps based on calculated cubic splines and linearly extrapolating any leading or trailing null values.

Examples

Natural cubic spline interpolation

Perform natural cubic spline interpolation:

```
SELECT CUBIC_SPLINE_APPROX(temperature, 'SPLINE_TYPE_NATURAL') OVER (PARTITION BY station ORDER BY ts) FROM WEATHER;
```

For natural boundary conditions, the BoundaryConditionArgument can be omitted. The following query returns identical results to the above one:

```
SELECT CUBIC_SPLINE_APPROX(temperature) OVER(PARTITION BY station ORDER BY ts) FROM WEATHER;
```
Perform cubic spline interpolation by using not-a-knot boundary conditions:

```sql
SELECT CUBIC_SPLINE_APPROX(temperature, 'SPLINE_TYPE_NOT_A_KNOT')
     OVER(PARTITION BY station ORDER BY ts) FROM WEATHER;
```

Cubic spline interpolation using series table

Perform cubic spline interpolation on a series by using a series table. In this example, the not-a-knot boundary condition is used, and it returns [?, 1, 2, 7, 10, 5, ?].

```sql
DROP TABLE InterpolationTable;
CREATE COLUMN TABLE InterpolationTable (ts_id VARCHAR(20), date DAYDATE, val DOUBLE)
    SERIES(SERIES KEY(ts_id) PERIOD FOR SERIES(date)
           EQUIDISTANT INCREMENT BY INTERVAL 1 DAY MISSING ELEMENTS ALLOWED);
INSERT INTO InterpolationTable VALUES('A','2013-09-29', null);
INSERT INTO InterpolationTable VALUES('A','2013-09-30', 1);
INSERT INTO InterpolationTable VALUES('A','2013-10-01', 2);
INSERT INTO InterpolationTable VALUES('A','2013-10-02', null);
INSERT INTO InterpolationTable VALUES('A','2013-10-03', 10);
INSERT INTO InterpolationTable VALUES('A','2013-10-04', 5);
INSERT INTO InterpolationTable VALUES('A','2013-10-05', null);
SELECT CUBIC_SPLINE_APPROX(val, 'SPLINE_TYPE_NOT_A_KNOT')
     OVER(SERIES TABLE InterpolationTable) AS MyResult
     FROM InterpolationTable;
```

**MYRESULT**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cubic spline interpolation together with linear extrapolation

This example extrapolates leading and trailing nulls, besides cubic spline interpolation. The slope of the line extrapolating the leading nulls is equal to the slope of the cubic spline at the first non-null value (which is '2013-09-29'). The slope of the line extrapolating the trailing nulls is equal to the slope of the cubic spline at the last non-null value (which is '2013-10-04'). This example refers to the InterpolationTable, as defined and populated in the previous example. This query returns [4, 1, 2, 7, 10, 5, -6].

```sql
SELECT date, val, CUBIC_SPLINE_APPROX(val, 'SPLINE_TYPE_NOT_A_KNOT')
     OVER(SERIES TABLE InterpolationTable) AS MyResult
     FROM InterpolationTable;
```

**DATE** | **VAL** | **MYRESULT**
----------|---------|---------
Sep 29, 2013 | ? | ?
### Linear extrapolation

Extrapolate leading and trailing nulls, besides cubic spline interpolation. The slope of the line extrapolating the leading nulls is equal to the slope of the cubic spline at the first non-null value ('2013-09-29'). The slope of the line extrapolating the trailing nulls is equal to the slope of the cubic spline at the last non-null value ('2013-10-04'). This example refers to the InterpolationTable as defined and populated in the previous example. This query returns [?, 1, 2, 7, 10, 5, ?].

```
SELECT date, val, CUBIC_SPLINE_APPROX(val, 'SPLINE_TYPE_NOT_A_KNOT') OVER(SERIES TABLE InterpolationTable) FROM InterpolationTable;
```

<table>
<thead>
<tr>
<th>DATE</th>
<th>VAL</th>
<th>MYRESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 30, 2013</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Oct 1, 2013</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Oct 2, 2013</td>
<td>?</td>
<td>7</td>
</tr>
<tr>
<td>Oct 3, 2013</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Oct 4, 2013</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Oct 5, 2013</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

This example performs linear extrapolation on a series.

```
SELECT date, val, LINEAR_APPROX(val, 'EXTRAPOLATION_LINEAR') OVER(SERIES(SERIES KEY(ts_id) PERIOD FOR SERIES(date) EQUIDISTANT INCREMENT BY INTERVAL 1 DAY MISSING ELEMENTS ALLOWED)) FROM InterpolationTable;
```

<table>
<thead>
<tr>
<th>DATE</th>
<th>VAL</th>
<th>LINEAR_APPROX(val,'EXTRAPOLATION_LINEAR')OVER(SERIES())</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 29, 2013</td>
<td>?</td>
<td>0</td>
</tr>
<tr>
<td>Sep 30, 2013</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Oct 1, 2013</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Oct 2, 2013</td>
<td>?</td>
<td>6</td>
</tr>
<tr>
<td>Oct 3, 2013</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Oct 4, 2013</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Oct 5, 2013</td>
<td>?</td>
<td>0</td>
</tr>
</tbody>
</table>

This example performs linear extrapolation on a series using a series table.

```
DROP TABLE InterpolationTable;
CREATE COLUMN TABLE InterpolationTable (ts_id VARCHAR(20), date DAYDATE, val DOUBLE) SERIES(SERIES KEY(ts_id) PERIOD FOR SERIES(date))
```

---

SAP HANA SQL Reference Guide for SAP HANA Platform

162 PUBLIC

SQL Reference
EQUIDISTANT INCREMENT BY INTERVAL 1 DAY MISSING ELEMENTS ALLOWED;
SELECT date, val, LINEAR_APPROX(val, 'EXTRAPOLATION_LINEAR') OVER(
    SERIES TABLE InterpolationTable) FROM InterpolationTable;

This example performs linear extrapolation on a series using a series table and returns \([1, 2, 6, 10]\).

DROP TABLE InterpolationTable;
CREATE COLUMN TABLE InterpolationTable
    (TS_ID VARCHAR(20), date DATE, val DOUBLE);
INSERT INTO InterpolationTable VALUES('A','2013-09-30', 1);
INSERT INTO InterpolationTable VALUES('A','2013-10-01', 2);
INSERT INTO InterpolationTable VALUES('A','2013-10-02', null);
INSERT INTO InterpolationTable VALUES('A','2013-10-03', 10);
SELECT date, val, LINEAR_APPROX(val, 'EXTRAPOLATION_LINEAR')
    OVER (PARTITION BY TS_ID ORDER BY date) FROM InterpolationTable;

<table>
<thead>
<tr>
<th>DATE</th>
<th>VAL</th>
<th>LINEAR_APPROX (VAL,'EXTRAPOLATION_LINEAR') OVER (SERIES())</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 30, 2013</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Oct 1, 2013</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Oct 2, 2013</td>
<td>?</td>
<td>6</td>
</tr>
<tr>
<td>Oct 3, 2013</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

If Value1Argument and Value2Argument are null or not specified, an input of \([\text{null, null, 1, null, 3, null, null}]\) returns \([-1, 0, 1, 2, 3, 4, 5]\). If the input contains only one non-null value, then all entries are replaced with copies of the non-null value. For example, an input of \([\text{null, null, 3, null}]\) returns \([3, 3, 3, 3]\).

If Value1Argument is set to 2 and Value2Argument is set to null then, an input of \([\text{null, null, 3, 4, 3, null, null}]\) returns \([2, 2, 3, 4, 3, 2, 2]\). If Value1Argument is set to 5 and Value2Argument is set to null and the input contains only one non-null value, then the higher value of Value1Argument and the non-null value from the input is used. For example, \([\text{null, null, 4, null, null}]\) returns \([5, 5, 4, 5, 5]\).

If Value1Argument is set to null and Value2Argument is set to 5, an input of \([\text{null, null, 4, 3, 4, null, null}]\) returns \([5, 5, 4, 3, 4, 5, 5]\). If Value1Argument is set to null and Value2Argument is set to 2 and the input contains only one non-null value, the lower value of Value2Argument and the non-null value from the input is used. For example, \([\text{null, null, 4, null, null}]\) returns \([2, 2, 4, 2, 2]\).

If Value1Argument is set to 2 and Value2Argument is set to 5, then an input of \([\text{null, null, 4, 3, null, null}]\) returns \([5, 5, 4, 3, 2, 2]\). If the input contains only one non-null value, the following conditions are checked.

- If Value1Argument is less than or equal to the non-null value, or if the non-null value is less than or equal to Value2Argument, then the non-null value is used.
- If the non-null value is less than Value1Argument, then Value1Argument is used.
- If the non-null value is greater than Value2Argument, then Value2Argument is used. For example, \([\text{null, null, 4, null, null}]\) returns \([3, 3, 4, 3, 3]\).

For example, \([\text{null, null, 4, null, null}]\) returns \([3, 3, 4, 3, 3]\).

If Value1Argument and Value2Argument are null or not specified, then any leading nulls are replaced by the first non-null value, and any trailing nulls are replaced by the last non-null value. For example, an input of \([\text{null, null, 1, 2, null, null}]\) returns \([1, 1, 1, 2, 2, 2]\). If the input only has one non-null value, then all the null values are replaced with the non-null value.
If Value1Argument is set to 5 and Value2Argument is set to null, then any leading nulls are replaced by Value1Argument, and the trailing nulls are replaced by the last non-null value. For example, an input of [null, null, 1, 2, null, null] returns [5, 5, 1, 2, 2, 2]. If the input only has one non-null value, then the leading nulls are replaced by Value1Argument and the trailing nulls are replaced by the non-null value. For example, an input of [null, null, 3, null] returns [5, 5, 3, 3]. If the input only has null values, then all null values are replaced with Value1Argument.

If Value1Argument is set to null and Value2Argument is set to 5, then any leading nulls are replaced by the first non-null value, and the trailing nulls are replaced by Value2Argument. For example, an input of [null, null, 1, 2, null, null] returns [1, 1, 1, 2, 5, 5]. If the input only has one non-null value, then the leading nulls are replaced by the non-null value and the trailing nulls are replaced by Value2Argument. For example, an input of [null, null, 3, null] returns [3, 3, 3, 5]. If the input only has null values, then all null values are replaced with Value2Argument.

If Value1Argument is set to 3 and Value2Argument is set to 4, any leading nulls are replaced by Value1Argument, and the trailing nulls are replaced by Value2Argument. For example, an input of [null, null, 1, 2, null, null] returns [3, 3, 1, 2, 4, 4]. If the input only has null values, then all null values are replaced with Value2Argument.

**Related Information**

- Expressions [page 56]
- Window Functions and the Window Specification [page 424]

### 4.9.1.51 CUME_DIST Function (Window)

Returns the relative rank of a row.

**Syntax**

```sql
CUME_DIST() <window_specification>
```

**Syntax Elements**

- `window_specification`
  
  Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].
Returns the relative rank of a row.

The relative rank of a row R is defined as NP/NR, where:

- NP is defined to be the number of rows preceding or peer with R in the window ordering of the window partition of R.
- NR is defined to be the number of rows in the window ordering of the window.

**Examples**

```
CREATE ROW TABLE ProductSales(ProdName VARCHAR(50), Description VARCHAR(20),
Sales INT);
INSERT INTO ProductSales VALUES('Tee Shirt','Plain',21);
INSERT INTO ProductSales VALUES('Tee Shirt','Lettered',22);
INSERT INTO ProductSales VALUES ('Tee Shirt','Team logo',30);
INSERT INTO ProductSales VALUES ('Hoodie','Plain',60);
INSERT INTO ProductSales VALUES ('Hoodie','Lettered',65);
INSERT INTO ProductSales VALUES ('Hoodie','Team logo',80);
INSERT INTO ProductSales VALUES('Ballcap','Plain',8);
INSERT INTO ProductSales VALUES('Ballcap','Lettered',40);
INSERT INTO ProductSales VALUES('Ballcap','Team logo',27);
SELECT ProdName, Description, Sales,
PERCENT_RANK() OVER (ORDER BY Sales ASC) AS Percent_Rank,
CUME_DIST() OVER (ORDER BY Sales ASC) AS Cume_Dist
FROM ProductSales
ORDER BY Sales DESC;
```
4.9.1.52 CURRENT_CONNECTION Function (Miscellaneous)

Returns the ID of the current connection.

Syntax

```sql
CURRENT_CONNECTION
```

Description

Returns the ID of the current connection as an integer value.

This function is called without braces.

Example

The following query returns the ID of the current connection.

```sql
SELECT CURRENT_CONNECTION "current connection" FROM DUMMY;
```

Related Information

M_CONNECTIONS System View [page 1785]
4.9.1.53 CURRENT_DATE Function (Datetime)

Returns the current local system date.

Syntax

```sql
CURRENT_DATE
```

Description

Returns the current local system date.

It is recommended that you use UTC dates instead of local dates. See to the CURRENT_UUTCDate function for more information.

Example

The following example returns the local and UTC date of the local system:

```sql
SELECT CURRENT_DATE "Current Date", CURRENT_UUTCDate "Coordinated Universal Date"
FROM DUMMY;
```

<table>
<thead>
<tr>
<th>Current Date</th>
<th>Coordinated Universal Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 1, 2019</td>
<td>May 1, 2019</td>
</tr>
</tbody>
</table>

```sql
SELECT CURRENT_DATE "Current Date", CURRENT_UUTCDate "Coordinated Universal Date"
FROM DUMMY;
```

<table>
<thead>
<tr>
<th>Current Date</th>
<th>Coordinated Universal Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 1, 2019</td>
<td>May 1, 2019</td>
</tr>
</tbody>
</table>

Related Information

Datetime Data Types [page 40]
CURRENT_UUTCDate Function (Datetime) [page 177]
4.9.1.54 CURRENT_IDENTITY_VALUE Function
(Miscellaneous)

Returns the latest inserted identity value in the current session.

Syntax Elements

CURRENT_IDENTITY_VALUE()

Description

Returns a BIGINT value representing the latest inserted identity value in the current session. If no identity value was inserted in the current session, then NULL is returned.

Example

The following example creates the table test and inserts a row with the identity 101. The identity value is then read and returned.

```
CREATE COLUMN TABLE test (objectid BIGINT GENERATED BY DEFAULT AS IDENTITY (START WITH 101), number INT);
INSERT INTO test (number) VALUES (1);
SELECT CURRENT_IDENTITY_VALUE() "current identity value" FROM DUMMY;
```

current identity value
101

4.9.1.55 CURRENT_MVCC_SNAPSHOT_TIMESTAMP Function
(Datetime)

Returns the timestamp of the current MVCC snapshot in SSSS format (seconds past midnight).

Syntax

CURRENT_MVCC_SNAPSHOT_TIMESTAMP()
**Description**

The CURRENT_MVCC_SNAPSHOT_TIMESTAMP function returns the timestamp of the current MVCC snapshot in SSSS format.

**Example**

The following example returns the timestamp of the current MVCC snapshot:

```
SELECT CURRENT_MVCC_SNAPSHOT_TIMESTAMP() FROM DUMMY;
```

**4.9.1.56 CURRENT_OBJECT_SCHEMA Function (Miscellaneous)**

Returns the current schema name when creating a view.

**Syntax**

```
CURRENT_OBJECT_SCHEMA()
```

**Description**

When creating a view, use the CURRENT_OBJECT_SCHEMA function where a schema name is required in the definition of the view, but a hard-coded schema name does not work. Currently, only the WORKDAYS_BETWEEN and CONVERT_CURRENCY functions support CURRENT_OBJECT_SCHEMA as replacement for an actual schema name.

**Example**

The following example replaces CURRENT_OBJECT_SCHEMA with the name of the current schema (in this case, SCHEMA1):

```
CREATE VIEW SCHEMA1.VIEW1 AS SELECT WORKDAYS_BETWEEN('01', '2014-01-09', '2014-01-10', CURRENT_OBJECT_SCHEMA()) FROM DUMMY;
```
4.9.1.57 CURRENT_SCHEMA Function (Miscellaneous)

Returns a string containing the current schema name.

Syntax

CURRENT_SCHEMA

Description

Returns a string containing the current schema name.

Example

The following example returns the current schema name:

SELECT CURRENT_SCHEMA "current schema" FROM DUMMY;

4.9.1.58 CURRENT_SITE_ID Function (Miscellaneous)

Returns the ID of the site that is calling the function. This function is primarily for use in SAP HANA System Replication (HSR).

Syntax

CURRENT_SITE_ID()

Description

Every site in an SAP HANA System Replication (HSR) environment has a site ID. Calling this function in an HSR system returns the ID of the site that is calling the function. Learning the site ID is useful when you are using forced routing.

When this function is called in a non-HSR system, 0 is returned.
Do not reference the DUMMY table in the FROM clause when calling the function. Instead, reference an actual table at the target location (for example, the `<sap_schema>svers` table, which is the recommended table for an SAP NetWeaver ABAP environment).

This function returns the current site ID once for every row in the table you reference (that is, the same site ID, but once for each table row). It is therefore recommended that you specify TOP 1 when calling the function to return the current site ID only once. See the example for how to do this.

**Example**

The following fictitious example returns the ID of the site that is executing the query:

```sql
SELECT TOP 1 CURRENT_SITE_ID() FROM mySchema.SVERS (RESULT_LAG('hana_sr'));
```

### 4.9.1.59 CURRENT_TIME Function (Datetime)

Returns the local system time.

**Syntax**

```sql
CURRENT_TIME
```

**Description**

Returns the current local system time.

It is recommended that you use UTC times instead of local times. See to the CURRENT_UTCTIME function for more information.

**Example**

The following example returns the local and UTC time of the system:

```sql
SELECT CURRENT_TIME "Current Time", CURRENT_UTCTIME "Coordinated Universal Time" FROM DUMMY;
```
Related Information

Datetime Data Types [page 40]
CURRENT_UTCTIME Function (Datetime) [page 178]

4.9.1.60  CURRENT_TIMESTAMP Function (Datetime)

Returns the current local system timestamp information.

Syntax

```
CURRENT_TIMESTAMP[(<precision>)]
```

Syntax Elements

- **precision**: Specifies the precision of the sub-seconds displayed. `<precision>` is an integer datatype with a valid range of 0-7. If not specified, the default precision is three.

Description

Returns the current local system timestamp information.

It is recommended that you use UTC timestamps instead of local timestamps. See the CURRENT_UTCTIMESTAMP function for more information.

Example

The following example returns the local timestamp of the system with zero, three, and seven precision values:

```
SELECT
  CURRENT_TIMESTAMP,
  CURRENT_TIMESTAMP(0),
  CURRENT_TIMESTAMP(7),
```
Related Information

Datetime Data Types [page 40]
CURRENT_UTCTIMESTAMP Function (Datetime) [page 179]

4.9.1.61 CURRENT_TRANSACTION_ISOLATION_LEVEL
Function (Miscellaneous)

Returns a string containing the current transaction isolation level.

Syntax

CURRENT_TRANSACTION_ISOLATION_LEVEL

Description

Returns a string containing the current transaction isolation level.

Example

The following example returns the current transaction isolation level:

```
SELECT CURRENT_TRANSACTION_ISOLATION_LEVEL "current transaction isolation level"
FROM DUMMY;
```
4.9.1.62 CURRENT_UPDATE_STATEMENT_SEQUENCE
Function (Miscellaneous)

Returns the number of write statements that have been issued in a transaction incremented by 1.

Syntax

CURRENT_UPDATE_STATEMENT_SEQUENCE()  

Description

Returns the number of write statements that have been issued in a transaction incremented by 1. If the transaction has never issued a write transaction, then the function returns 1. The initial value is 1. In a read transaction, the function always returns 1.

Examples

The following example shows how to retrieve the current write statement sequence number of the current transaction and returns 1:

```
CREATE COLUMN TABLE T(id INT);
INSERT INTO T VALUES (1);
SELECT CURRENT_UPDATE_STATEMENT_SEQUENCE() "statement sequence number" FROM DUMMY;
```

The following statement also returns 1:

```
DELETE FROM T;
SELECT CURRENT_UPDATE_STATEMENT_SEQUENCE() "statement sequence number" FROM DUMMY;
```
4.9.1.63 CURRENT_UPDATE_TRANSACTION Function (Miscellaneous)

Returns the unique ID of the current transaction when it is in write mode.

Syntax

CURRENT_UPDATE_TRANSACTION()

Description

Returns the unique ID (BIGINT) of the current transaction when it is in write mode. If the current transaction is in read mode, the function returns 0.

Example

The following example returns the current update transaction id.

```
SELECT CURRENT_UPDATE_TRANSACTION() "current update transaction" FROM DUMMY;
```

4.9.1.64 CURRENT_USER Function (Miscellaneous)

Returns the current user name at the current statement context.

Syntax

CURRENT_USER

Description

Returns the current user name at the current statement context. This is the user name that is currently at the top of authorization stack.
SESSION_USER and CURRENT_USER are similar functions, but there are differences. SESSION_USER reflects how the session connected to the server, while CURRENT_USER may be affected by the SQL SECURITY clause and reflects how permissions are applied.

Example

The following example shows a basic function operation using the SYSTEM user and returns SYSTEM:

```sql
SELECT CURRENT_USER "current user" FROM DUMMY;
```

In the following example, USER A creates a procedure:

```sql
CREATE PROCEDURE USER_A.PROC1 LANGUAGE SQLSCRIPT SQL SECURITY DEFINER AS
BEGIN
    SELECT CURRENT_USER "current user" FROM DUMMY;
END;
```

USER B calls the procedure USER_A.PROC1, which returns USER_A:

```sql
CALL USER_A.PROC1;
```

USER A also creates the following procedure:

```sql
CREATE PROCEDURE USER_A.PROC2 LANGUAGE SQLSCRIPT SQL SECURITY INVOKER AS
BEGIN
    SELECT CURRENT_USER "current user" FROM DUMMY;
END;
```

USER B calls the procedure USER_A.PROC2, which returns USER_B:

```sql
CALL USER_A.PROC2;
```

Related Information

SESSION_USER Function (Miscellaneous) [page 342]

4.9.1.65 CURRENT_USER_ID Function (Miscellaneous)

Returns the current user’s user ID at the current statement context.

Syntax

```sql
CURRENT_USER_ID()
```
**Description**

Returns the current user’s user ID at the current statement context. This is the user’s ID that is currently at the top of the authorization stack.

**Example**

```
SELECT CURRENT_USER_ID() from DUMMY;
```

### 4.9.1.66 CURRENT_UTCDATE Function (Datetime)

Returns the current UTC date.

**Syntax**

```
CURRENT_UTCDATE
```

**Description**

Returns the current UTC date.

In application code, it is recommended that you use UTC dates instead of local dates.

**Example**

The following example returns the local and UTC date of the local system:

```
SELECT CURRENT_DATE "Current Date", CURRENT_UTCDATE "Coordinated Universal Date"
FROM DUMMY;
```

<table>
<thead>
<tr>
<th>Current Date</th>
<th>Coordinated Universal Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>May 1, 2019</td>
</tr>
</tbody>
</table>
4.9.1.67  CURRENT_UTCTIME Function (Datetime)

Returns the current UTC time.

Syntax

```sql
CURRENT_UTCTIME
```

Description

Returns the current UTC time.

Example

The following example returns the local and UTC time of the system:

```sql
SELECT CURRENT_TIME "Current Time", CURRENT_UTCTIME "Coordinated Universal Time"
FROM DUMMY;
```

<table>
<thead>
<tr>
<th>Current Time</th>
<th>Coordinated Universal Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:56:51 AM</td>
<td>2:56:51 PM</td>
</tr>
</tbody>
</table>

Related Information

Datetime Data Types [page 40]
4.9.1.68 CURRENT_UTCTIMESTAMP Function (Datetime)

Returns the current UTC timestamp.

Syntax

```
CURRENT_UTCTIMESTAMP[(<precision>)]
```

Syntax Elements

- `<precision>` Specifies the precision of the sub-seconds displayed. `<precision>` is an integer datatype with a valid range of 0-7. If not specified, the default precision is three.

Description

Returns the current UTC timestamp.

Example

The following example returns the UTC timestamp of the system with zero, three, and seven precision values:

```
SELECT     CURRENT_UTCTIMESTAMP
           CURRENT_UTCTIMESTAMP(0)
           CURRENT_UTCTIMESTAMP(3)
           CURRENT_UTCTIMESTAMP(7)
FROM DUMMY;
```

'2021-03-09 09:07:36.8110000',
'2021-03-09 09:07:36.0000000',
'2021-03-09 09:07:36.8110000',
'2021-03-09 09:07:36.8118739'

Related Information

Datetime Data Types [page 40]
CURRENT_TIMESTAMP Function (Datetime) [page 172]
4.9.1.69  **DAYNAME Function (Datetime)**

Returns the weekday name for the specified date.

**Syntax**

```
DAYNAME(<date>)
```

**Description**

Returns the weekday in English for the specified date.

**Example**

The following example returns Monday as the weekday for the specified date:

```
SELECT DAYNAME ('2011-05-30') "dayname" FROM DUMMY;
```

**Related Information**

[Datetime Data Types](#) [page 40]

4.9.1.70  **DAYOFMONTH Function (Datetime)**

Returns the day of the month for the specified date.

**Syntax**

```
DAYOFMONTH(<date>)
```
**Description**

Returns an integer for the day of the month for the specified date.

**Example**

The following example returns 30 as the number for the day of the month for the specified date:

```
SELECT DAYOFMONTH ('2011-05-30') "dayofmonth" FROM DUMMY;
```

**Related Information**

Datetime Data Types [page 40]

---

4.9.1.71 DAYOFYEAR Function (Datetime)

Returns an integer representation of the day of the year for the specified date.

**Syntax**

```
DAYOFYEAR(<date>)
```

**Description**

Returns an integer representation of the day of the year for the specified date.

**Example**

The following example returns the value 150 as the day of the year for the specified date:

```
SELECT DAYOFYEAR ('2011-05-30') "dayofyear" FROM DUMMY;
```
4.9.1.72 DAYS_BETWEEN Function (Datetime)

Computes the number of entire (24 hour) days between two dates.

Syntax

```
DAYS_BETWEEN(<date_1>, <date_2>)
```

Syntax Elements

- `date_1`
  Specifies the starting TIMESTAMP for the comparison.
- `date_2`
  Specifies the ending TIMESTAMP for the comparison.

Description

Computes the number of entire days between `<date_1>` and `<date_2>`.

Example

The following example returns the value 31 for days between the two dates specified:

```
SELECT DAYS_BETWEEN (TO_DATE ('2009-12-05', 'YYYY-MM-DD'), TO_DATE('2010-01-05', 'YYYY-MM-DD')) "days between" FROM DUMMY;
```

The following example returns the value 0 for days between the two specified dates:

```
SELECT DAYS_BETWEEN('2018-02-07 23:00:00', '2018-02-08 01:00:00') AS sinceDays FROM dummy;
```
The following example returns the value 1 for days between the two specified dates:

```
SELECT DAYS_BETWEEN('2018-02-07 23:00:00', '2018-02-08 23:00:00') AS sinceDays
FROM dummy;
```

Related Information

Datetime Data Types [page 40]
SAP Note 2573900 - Changed Behavior of the DAYS_BETWEEN function in HANA 2.0 SPS03

4.9.1.73 DENSE_RANK Function (Window)

Performs the same ranking operation as the RANK function, except that rank numbering does not skip when ties are found.

Syntax

```
DENSE_RANK() <window_specification>
```

Syntax Elements

```
window_specification
```

Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].

Description

The DENSE_RANK function performs the same ranking operation as the RANK function, except that rank numbering does not skip when ties are found.
Examples

In this example, the RANK function returns the rank 5 for the row ('A', 10, 0) because there were two rows that returned the rank of 3. However, DENSE_RANK returns the rank 4 for the row ('A', 10, 0).

```
CREATE ROW TABLE T (class CHAR(10), val INT, offset INT);
INSERT INTO T VALUES('A', 1, 1);
INSERT INTO T VALUES('A', 3, 3);
INSERT INTO T VALUES('A', 5, null);
INSERT INTO T VALUES('A', 5, 2);
INSERT INTO T VALUES('A', 10, 0);
INSERT INTO T VALUES('B', 1, 3);
INSERT INTO T VALUES('B', 1, 1);
INSERT INTO T VALUES('B', 7, 1);
SELECT class,
    val,
    ROW_NUMBER() OVER (PARTITION BY class ORDER BY val) AS row_num,
    RANK() OVER (PARTITION BY class ORDER BY val) AS rank,
    DENSE_RANK() OVER (PARTITION BY class ORDER BY val) AS dense_rank
FROM T;
```

<table>
<thead>
<tr>
<th>CLASS</th>
<th>VAL</th>
<th>ROW_NUM</th>
<th>RANK</th>
<th>DENSE_RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Related Information

Expressions [page 56]
Window Functions and the Window Specification [page 424]
4.9.1.74 DFT Function (Aggregate)

Computes a discrete Fourier transform of series.

Syntax

```
DFT( <expression>, <N> { <series_orderby> | <order_by_clause> } ) { REAL | IMAGINARY | AMPLITUDE | PHASE }
```

Syntax Elements

**expression**

Specifies a value assumed to be a sample taken at a constant time interval. `<expression>` cannot contain any NULL values.

**N**

This parameter must be a power of 2. The input is padded with zeros if it contains less than `<N>` elements.

**series_orderby**

The SERIES definition can only be used with an equidistant series.

```
<series_orderby> ::= SERIES( <series_period> <series_equidistant_definition> )
```

The series must not contain missing elements. For more information about this clause, see the SERIES_GENERATE function.

**order_by_clause**

Specifies the sort order of the input rows.

```
<order_by_clause> ::= ORDER BY <order_by_expression> [ , <order_by_expression> [...] ]

<order_by_expression> ::= <column_name> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [ <collate_clause> ]
| <column_position> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [ <collate_clause> ]
<collate_clause> ::= COLLATE <collation_name>
```

`<collate_clause>` specifies the collation to use for ordering values in the results. `<collate_clause>` can only be used on columns defined as NVARCHAR or VARCHAR. `<collation_name>` is one of the supported collation names listed in the COLLATIONS system view.
Description

Computes the Discrete Fourier Transform of an expression for the first \(<N>\) values and returns an array with exactly \(<N>\) elements.

The returned values depend on the output parameter, which must be one of REAL, IMAGINARY, AMPLITUDE, or PHASE.

Examples

The example below computes the Discrete Fourier Transform of a column in an equidistant series.

```sql
SELECT DFT(FRACTION_OF_MIN_MAX_RANGE, 4 SERIES(EQUIDISTANT INCREMENT BY 1 PERIOD FOR SERIES(element_number))).REAL
FROM SERIES_GENERATE_INTEGER(1,0,10);
```

The example below uses the fictional MY_TABLE and returns an array with 4 numbers representing the real part of the result.

```sql
SELECT DFT(col, 4 ORDER BY DATE).REAL FROM MY_TABLE;
```

The example below uses the fictional MY_TABLE and returns an array with 8 numbers representing the imaginary part of the result.

```sql
SELECT DFT(col, 8 ORDER BY DATE).IMAGINARY FROM MY_TABLE;
```

The example below uses the fictional MY_TABLE and returns an array with 8 numbers representing the amplitude part (i.e. SQRT(REAL^2 + IMAGINARY^2)) of the result.

```sql
SELECT DFT(col, 8 ORDER BY DATE).AMPLITUDE FROM MY_TABLE;
```

The example below uses the fictional MY_TABLE returns an array with 8 numbers representing the phase part of the result and ranges between -PI and +PI.

```sql
SELECT DFT(col, 8 ORDER BY DATE).PHASE FROM MY_TABLE;
```

Related Information

- Aggregate Functions [page 414]
- COLLATIONS System View [page 1520]
- SERIES_GENERATE Function (Series Data) [page 331]
4.9.1.75 ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS
Function (Security)

Extracts root keys and sends them to a client session as a CLOB.

Syntax

```
ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS(['<root_keytype_list>'])
```

Syntax Elements

`root_keytype_list`

Specify the key types that are included in the CLOB. All versions of the specified key type are included in the CLOB. If `<root_keytype_list>` is empty, then all key types are included in the CLOB.

```
<root_keytype_list> ::=  <root_keytype> [, ...]  <root_keytype> ::=  PERSISTENCE | APPLICATION | LOG | BACKUP
```

- **PERSISTENCE** The following statement extracts all persistence and log root keys from the encryption root.
  Specifies that persistence encryption root keys are extracted.
- **APPLICATION** Specifies that application encryption root keys are extracted.
- **LOG** Specifies that log encryption root keys are extracted.
- **BACKUP** Specifies that backup encryption root keys are extracted.

Description

Execute this statement in the database from which the root keys are being extracted.

Permissions

You must have the ENCRYPTION ROOT KEY ADMIN privilege and the encryption root key backup password should be set in the encryption root key store (instance SSFS or local secure store (LSS)).
Example

The following statement extracts all persistence and log root keys from the encryption root key store (instance SSFS) and sends them to a client session as a CLOB.

```
SELECT ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS ('PERSISTENCE, LOG') FROM DUMMY;
```

The following statement includes all key types in the CLOB.

```
SELECT ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS ('') FROM DUMMY;
```

Related Information

ALTER SYSTEM VALIDATE ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 587]
ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
M_PERSISTENCE_ENCRYPTION_KEYS System View [page 2042]
Server-Side Data Encryption Services
Managing Data Encryption in SAP HANA
ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 568]
ALTER SYSTEM LOG ENCRYPTION Statement (System Management) [page 541]
ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
ENCRYPTION_ROOT_KEYS System View [page 1558]
APPLICATION_ENCRYPTION_KEYS System View [page 1498]
M_ENCRYPTION_OVERVIEW System View [page 1881]

4.9.1.76 ENCRYPTION_ROOT_KEYS_EXTRACT_ALL_KEYS_FOR_DATABASE Function (Security)

Extracts all the encryption root keys of a database and sends them to a client session as a CLOB.

Syntax

```
ENCRYPTION_ROOT_KEYS_EXTRACT_ALL_KEYS_FOR_DATABASE ('<database_name>')
```
Syntax Elements

database_name

Specifies the name of the database whose root keys are included in the CLOB. All versions of all key types are included in the CLOB.

Description

The root keys of a tenant database can be extracted using this function either in the tenant database (in which case the database name is still mandatory), or in the system database if the system database has encryption configuration control for the tenant database. The returned root key backup is base64 encoded and therefore compatible to the CLOB data type. It always contains all root keys for all root key types (that is: DPAPI, BACKUP, LOG, PERSISTENCE).

The returned backup file format depends on the used secure store (local secure store or secure store in the file system).

Permissions

You must have the ENCRYPTION ROOT KEY ADMIN system privilege and the encryption root key backup password must be set.

Example

The following statement extracts all root keys from the key store and sends them to a client session as a CLOB.

```
SELECT ENCRYPTION_ROOT_KEYS_EXTRACT_ALL_KEYS_FOR_DATABASE ('my_database') FROM DUMMY;
```

Related Information

ALTER SYSTEM VALIDATE ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 587]
ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 568]
M_PERSISTENCE_ENCRYPTION_KEYS System View [page 2042]
ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
ALTER SYSTEM LOG ENCRYPTION Statement (System Management) [page 541]
ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
4.9.1.77 ENCRYPTION_ROOT_KEYS_HAS_BACKUP_PASSWORD Function (Security)

Returns a value that indicates whether the root key backup password is set.

Syntax

ENCRYPTION_ROOT_KEYS_HAS_BACKUP_PASSWORD()

Description

Returns 1 if the password is already set or 0 if there is no password.

Related Information

Back Up Root Keys
ALTER SYSTEM APPLICATION ENCRYPTION Statement (System Management) [page 512]
ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
ALTER SYSTEM LOG ENCRYPTION Statement (System Management) [page 541]
ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
4.9.1.78  ESCAPE_DOUBLE_QUOTES Function (Security)

Escapes double quotes in the specified string.

Syntax

```
ESCAPE_DOUBLE_QUOTES(<value>)
```

Description

Escapes double quotes in the `<value>` string, ensuring that a valid SQL identifier is used in dynamic SQL statements to prevent SQL injections. The function returns the input string with escaped double quotes.

Example

The following query escapes the double quotes and returns the value `TAB"LE`.

```
SELECT ESCAPE_DOUBLE_QUOTES('TAB"LE') "table_name" FROM DUMMY;
```

4.9.1.79  ESCAPE_SINGLE_QUOTES Function (Security)

Escapes single quotes in the specified string.

Syntax

```
ESCAPE_SINGLE_QUOTES(<value>)
```

Description

Escapes single quotes (apostrophes) in the given string `<value>`, ensuring a valid SQL string literal is used in dynamic SQL statements to prevent SQL injections. Returns the input string with escaped single quotes.
Examples

The following query escapes the parameter content `Str'ing` to `Str''ing`.

```
SELECT ESCAPE_SINGLE_QUOTES('Str''ing') "string_literal" FROM DUMMY;
```

The following query example shows the strings retrieved from a table `t`, both without and with ESCAPE_SINGLE_QUOTES applied. The column `col_txt` contains the two entries `Adam's` and `Eve`.

```
CREATE COLUMN TABLE txt (  
  col_txt NVARCHAR(5000) NOT NULL);
INSERT INTO txt VALUES ('Adam''s');  
INSERT INTO txt VALUES ('Eve');  
SELECT col_txt, escape_single_quotes(col_txt) FROM txt;
```

The results are as follows:

<table>
<thead>
<tr>
<th>COL_TXT</th>
<th>ESCAPE_SINGLE_QUOTES(COL_TXT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam's</td>
<td>Adam''s</td>
</tr>
<tr>
<td>Eve</td>
<td>Eve</td>
</tr>
</tbody>
</table>

4.9.1.80  EXP Function (Numeric)

Returns the result of the base of the natural logarithms e raised to the power of the specified argument.

Syntax

```
EXP(<number>)
```

Description

Returns the result of the base of the natural logarithms e raised to the power of the argument `<number>`.

Example

The following example returns the value `2.718281828459045` for `"exp"`:

```
SELECT EXP (1.0) "exp" FROM DUMMY;
```
4.9.1.81 EXPRESSION_MACRO Function (Miscellaneous)

Returns aggregated results from a query.

Syntax

EXPR_ESSION_MACRO( <expression_macro_alias> )

Syntax Elements

expression_macro_alias Specifies the name of an expression macro.

Description

Use the EXPRESSION_MACRO function to select an expression that is based on columns and other EXPRESSION_MACROs from the view and that may also involve aggregation functions (like AVG). A SELECT statement with an EXPRESSION_MACRO that involves an aggregation function requires the same GROUP BY clause as if the aggregation function had appeared directly in the SELECT statement.

The list of all expression macros can be found in the VIEW_EXPRESSION_MACROS system view.

Example

The following example creates a table, defines a view on the table that includes an expression macro, and then queries the view using the EXPRESSION_MACRO function to have the expression macro perform calculations on the result. In this example, the value 15 is returned (the average of 10 and 20).

```
CREATE TABLE t1(a INT);
INSERT INTO t1 VALUES(10);
INSERT INTO t1 VALUES(20);
CREATE VIEW v1 AS SELECT * FROM t1 WITH EXPRESSION MACROS(AVG(a) AS avgA);
SELECT EXPRESSION_MACRO(avgA) FROM v1;
```
4.9.1.82 EXTRACT Function (Datetime)

Returns the requested portion of a specified date.

Syntax

```
EXTRACT( {YEAR | MONTH | DAY | HOUR | MINUTE | SECOND} FROM <date> )
```

Description

Returns the requested portion from the specified date.

Example

The following example returns the value 2010 for the year extracted from the specified date:

```
SELECT EXTRACT (YEAR FROM TO_DATE ('2010-01-04', 'YYYY-MM-DD')) "extract" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]
4.9.1.83 FIRST_VALUE Function (Aggregate)

Returns the value of the first element of an expression. This function can also be used as a window function.

Syntax

Aggregate function:

```
FIRST_VALUE( <expression> <order_by_clause> )
```

Window function:

```
FIRST_VALUE( <expression> <order_by_clause> ) <window_specification>
```

Syntax Elements

- **expression**
  Specifies the expression of data to operate over.

- **order_by_clause**
  Specifies the sort order of the input rows.

```
<order_by_clause> ::= ORDER BY <order_by_expression> [, <order_by_expression> [
[,] ... ]]
<order_by_expression> ::=<column_name> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [ <collate_clause> ]
| <column_position> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [ <collate_clause> ]
<collate_clause> ::= COLLATE <collation_name>
```

- **<collate_clause>** specifies the collation to use for ordering values in the results. **<collate_clause>** can only be used on columns defined as NVARCHAR or VARCHAR. **<collation_name>** is one of the supported collation names listed in the COLLATIONS system view.

- **window_specification**
  Defines a window on the data over which the function operates. For **<window_specification>**, see Window Functions and the Window Specification [page 424].

Description

Returns the value of the first element in **<expression>** as ordered by **<order_by_clause>**.

NULL is returned if the value is NULL or if **<expression>** is empty.

The output of the FIRST_VALUE function can be non-deterministic among tie values.
Example

The example below returns the first value in the COL1 column when the table is ordered by COL2:

```
CREATE ROW TABLE T (COL1 DOUBLE, COL2 DOUBLE);
INSERT INTO T VALUES(9, 1);
INSERT INTO T VALUES(4, 5);
INSERT INTO T VALUES(7, 3);
SELECT FIRST_VALUE (COL1 ORDER BY COL2) FROM T;
```

The query returns 9.

Example for Spatial Data Type (ST_Geometry, ST_Point)

The example below returns the first value, POINT (1 1), in binary format, when the table is ordered by ID:

```
CREATE TABLE TAB (ID INT, SHAPE ST_Geometry(4326));
INSERT INTO TAB VALUES(1, ST_GeomFromText('POINT(1 1)', 4326));
INSERT INTO TAB VALUES(2, ST_GeomFromText('POINT(2 2)', 4326));
INSERT INTO TAB VALUES(3, ST_GeomFromText('POINT(3 3)', 4326));
INSERT INTO TAB VALUES(4, ST_GeomFromText('POINT(4 4)', 4326));
SELECT FIRST_VALUE(SHAPE ORDER BY ID) FROM TAB;
```

To return a result as data type NVARCHAR, use the following statement:

```
SELECT FIRST_VALUE(CAST(SHAPE AS NVARCHAR) ORDER BY ID) FROM TAB;
```

Related Information

COLLATIONS System View [page 1520]
Window Functions and the Window Specification [page 424]
Window Aggregate Functions [page 428]
Aggregate Functions [page 414]
SAP HANA Spatial Reference

4.9.1.84  FLOOR Function (Numeric)

Returns the largest integer that is not greater than the specified numeric argument.

Syntax

```
FLOOR(<number>)
```
Description

Returns the largest decimal value that is not greater than the numeric argument <number>.

Example

The following example returns the value 14.0 for "floor":

```
SELECT FLOOR (14.5) "floor" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.85 GENERATE_PASSWORD Function (Security)

Generates a password.

Syntax

```
GENERATE_PASSWORD( <password_length> [, <usergroup_name>] )
```

Syntax Elements

- **password_length**
  - Specifies the length of the password to generate as an integer.
- **usergroup_name**
  - Specifies a usergroup name. If there are password policy parameters defined for the usergroup, they are applied during password generation.

Description

Generates and returns a password of type VARCHAR(128).
iNote

The function call can fail displaying the following out of range error message if the password policy and exclude list are too restrictive:

password exclude list too restrictive

Depending on the strictness of the password exclude list, it might suffice to retry the command.

Example

The following example returns a generated password of 16 characters:

```
SELECT GENERATE_PASSWORD(16) FROM Dummy;
```

4.9.1.86 GREATEST Function (Miscellaneous)

Returns the greatest value among the specified arguments.

Syntax

```
GREATEST(<argument> [{, <argument>}...])
```

Description

Returns the greatest value among the arguments (n1, n2,...). If one of the arguments is NULL, the return is NULL.

Example

The following example returns bb as the greatest value:

```
SELECT GREATEST ('aa', 'ab', 'ba', 'bb') "greatest" FROM DUMMY;
```
4.9.1.87 GROUP_SCORE Function (Miscellaneous)

Returns the relevance of grouped records found.

Syntax

GROUP_SCORE ( <search_string> IN <score_columns> )

Syntax Elements

score_columns

<score_columns> ::=      { <single_column>     | <specific_columns_diff_options>     | <specific_columns_same_options>     | <all_columns> }

single_column

<single_column> ::=      <column_name> [ <scoring_type> ]

scoring_type

<scoring_type> ::=      { EXACT [ <exact_options> ]     | FUZZY [ <fuzzy_options> ]     | LINEAR <scoring_details>     | GAUSSIAN <scoring_details>     | LOGARITHMIC <logarithmic_scoring_details> }

exact_options

<exact_options> ::=      { ABBREVIATION SIMILARITY <value_range> AND SYMMETRIC ('ON' | 'OFF') AND THRESHOLD <value_range> MINIMAL SCORE <value_range> WEIGHT <value_range_2> COMPOUND WORD WEIGHT <value_range_2> NON MATCHING TOKEN MODE ('max' | 'min' | 'all' | 'input' | 'table') DECOMPOSE WORDS <positive_integer_constant> EXCESS TOKEN WEIGHT <value_range> PHRASE CHECK FACTOR <value_range> SEARCH MODE <search_mode> }

fuzzy_search

<fuzzy_options> ::=      { ABBREVIATION SIMILARITY <value_range> AND SYMMETRIC ('ON' | 'OFF') AND THRESHOLD <value_range> }
AND THRESHOLD <value_range>
MINIMAL SCORE <value_range>
WEIGHT <value_range_2>
COMPOSE WORDS <positive_integer_constant>
COMPOUND WORD WEIGHT <value_range_2>
NON MATCHING TOKEN MODE {'max' | 'min' | 'all' | 'input' | 'table'}
DECOMPOSE WORDS <positive_integer_constant>
ERROR DEvaluate <value_range>
EXCESS TOKEN WEIGHT <value_range>
SUBSTRING MATCH {'off' | 'anywhere' | 'beginning'}
LANGUAGE TRANSLATION {'on' | 'off'}
LENGTH TOLERANCE <value_range>
MAX DATE DISTANCE {0 | <positive_integer_constant>}
MINIMAL SEARCH LENGTH {0 | <positive_integer_constant>}
PHRASE CHECK FACTOR <value_range>
SEARCH MODE <search_mode>
SIMILARITY CALCULATION MODE <similar_calculation_mode>
SPELL CHECK FACTOR <value_range>
MINIMAL TOKEN SCORE <value_range>

value_range_1 A value that is greater than or equal to 0.0 and less than or equal to 1. For example, 0.326, 0.57, or 0.9.
value_range_2 A value that is greater than 0 and less than 1.0.
positive_float_constant A positive integer greater than 1.0. For example, 57.326.
positive_integer_constant A positive integer constant. For example, 5.
float_constant A non-negative value.

search_mode

<search_mode> ::=  
  {'alphanum'
   'alphanum identifier'
   'house number'
   'identifier'
   'postcode'
   'string'
   'text'}

similar_calculation_mode

<similar_calculation_mode> ::=  
  {'compare'
   'type ahead'
   'search compare'
   'search'
   'symmetric search'
   'substring search'
   'flexible'}

scoring_details

<scoring_details> ::=  
  [ MINIMAL SCORE <value_range> ]  
  [ OFFSET {0 | <positive_float_constant>} ]  
  [ DECAY <decay_value> ]  
  SCALE <positive_float_constant>  
  [ SEACH MODE {'date' | 'numc'} ]  
  [ WEIGHT <weight_value> ]

decay_value A value no less than 0.0 but less than 1.0.
weight_value A value greater than 0.0 and less than or equal to 1.0.

logarithmic_scoring_details

<logarithmic_scoring_details> ::=
Description

Use GROUP_SCORE to obtain the relevance of a group of records that have been found. GROUP_SCORE returns a real value between 0 and 1. The SAP HANA database calculates a score based on the following information:

- The relevance, or weighting, of attributes. The relevance of a match depends on the weight of the column that caused the match. You can specify weights as you create the view.
- Fuzziness in a fuzzy search. The more exact matching that occurs, the higher the score is.
- Text ranking (TF-IDF).

Examples

The following example creates two table that contains several strings:

```sql
CREATE TABLE companies (id int, name varchar(255));
INSERT INTO companies VALUES (1, 'Company One');
INSERT INTO companies VALUES (2, 'Company Two');
INSERT INTO companies VALUES (3, 'Company Three');
CREATE TABLE locations (company_id int, location nvarchar(255));
INSERT INTO locations VALUES (1, 'Berlin');
INSERT INTO locations VALUES (1, 'Seattle');
INSERT INTO locations VALUES (1, 'Tokyo');
INSERT INTO locations VALUES (2, 'Berlin');
INSERT INTO locations VALUES (2, 'Tokyo');
INSERT INTO locations VALUES (3, 'London');
```

Use the GROUP_SCORE function to check the data for relevance against the search string 'Berlin Seattle Tokyo':

```sql
SELECT name, group_score('Berlin Seattle Tokyo' IN (name, location))
FROM companies JOIN locations ON id = company_id
```
GROUP BY name;

<table>
<thead>
<tr>
<th>NAME</th>
<th>group_score('Berlin Seattle Tokyo' IN (name, location))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company One</td>
<td>1.0</td>
</tr>
<tr>
<td>Company Two</td>
<td>0.42264974</td>
</tr>
<tr>
<td>Company Three</td>
<td>0.0</td>
</tr>
</tbody>
</table>

### 4.9.1.88 GROUPING Function (Miscellaneous)

Determines whether a specified column is used in grouping.

**Syntax**

GROUPING(<column_name>)

**Description**

The GROUPING function can be used with GROUPING SETS, ROLLUP, or CUBE, which return multiple levels of aggregation in a single result set. The function returns 1 if the specified column is used in aggregation, and 0 otherwise. The column of GROUPING must be an element of the GROUPING SETS.

**Example**

```sql
CREATE COLUMN TABLE mySchema.Customers (  
cust_id INTEGER NOT NULL,
cust_name NVARCHAR(20),
um_emp INTEGER,
region NVARCHAR(20),
s_tier NVARCHAR(20),
PRIMARY KEY ("CUST_ID")
);

INSERT INTO mySchema.Customers VALUES(  
1, 'CustA', 5, 'NorthEast', 'gold' );
INSERT INTO mySchema.Customers VALUES(  
2, 'CustB', 26, 'NorthWest', 'gold' );
INSERT INTO mySchema.Customers VALUES(  
3, 'CustC', 250, 'NorthEast', 'silver' );
INSERT INTO mySchema.Customers VALUES(  
4, 'CustD', 180, 'SouthEast',  
'platinum' );
INSERT INTO mySchema.Customers VALUES(  
5, 'CustE', 32, 'SouthWest', 'silver' );
INSERT INTO mySchema.Customers VALUES(  
6, 'CustF', 45, 'NorthEast', 'platinum' );
INSERT INTO mySchema.Customers VALUES(  
7, 'CustG', 15, 'NorthWest', 'platinum' );
INSERT INTO mySchema.Customers VALUES(  
8, 'CustH', 99, 'SouthEast', 'gold' );
INSERT INTO mySchema.Customers VALUES(  
9, 'CustI', 6, 'NorthEast', 'silver' );
INSERT INTO mySchema.Customers VALUES(  
10,'CustJ', 101, 'NorthEast', 'silver' );
```

SAP HANA SQL Reference Guide for SAP HANA Platform

**SQL Reference**
```sql
INSERT INTO mySchema.Customers VALUES( 11, 'Custk', 108, 'SouthEast', 'silver' );
SELECT cust_name AS "cust_name", cust_id AS "cust_id", region AS "region", s_tier AS "s_tier", num_emp AS "num_emp", GROUPING (region) AS "gr_reg", GROUPING (s_tier) AS "gr_tier", GROUPING (num_emp) AS "gr_num"
FROM mySchema.Customers
GROUP BY GROUPING SETS ( (s_tier, region), (region, s_tier), (cust_id, cust_name, num_emp) );
```

<table>
<thead>
<tr>
<th>cust_name</th>
<th>cust_id</th>
<th>region</th>
<th>sales_tier</th>
<th>num_emp</th>
<th>gr_reg</th>
<th>gr_tier</th>
<th>gr_num</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NorthEast</td>
<td>gold</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NorthWest</td>
<td>gold</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>SouthEast</td>
<td>gold</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NorthEast</td>
<td>platinum</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NorthWest</td>
<td>platinum</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>SouthEast</td>
<td>platinum</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NorthEast</td>
<td>silver</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>SouthEast</td>
<td>silver</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>SouthWest</td>
<td>silver</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NorthEast</td>
<td>gold</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NorthWest</td>
<td>gold</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>SouthEast</td>
<td>gold</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NorthEast</td>
<td>platinum</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>SouthEast</td>
<td>platinum</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NorthEast</td>
<td>silver</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>SouthEast</td>
<td>silver</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CustA</td>
<td>1</td>
<td>NULL</td>
<td>NULL</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CustB</td>
<td>9</td>
<td>NULL</td>
<td>NULL</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CustC</td>
<td>7</td>
<td>NULL</td>
<td>NULL</td>
<td>15</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
### 4.9.1.89 GROUPING_ID Function (Miscellaneous)

Returns an integer value to identify which grouping set each row belongs to.

**Syntax**

```
GROUPING_ID(<column_name_list>)
```

**Description**

The GROUPING_ID function can be used with GROUPING SETS, ROLLUP, or CUBE, which return multiple levels of aggregation in a single result set. The GROUPING_ID function returns an integer value to identify which grouping set each row belongs to. Each column in GROUPING_ID must be an element of GROUPING_SETS.

GROUPING_ID is assigned by converting the bit vector generated from GROUPING SETS to a decimal number by treating the bit vector as a binary number. When a bit vector is composed, 0 is assigned to each column specified in the GROUPING SETS and 1 otherwise in the order it appears in GROUPING SETS. By treating the bit vector as a binary number, this function returns an integer value as the output.
The following statement uses the fictitious table guided_navi_tab and the results indicates which grouping set each row belongs to:

```sql
SELECT customer, year, product, SUM(sales), GROUPING_ID(customer, year, product)
FROM guided_navi_tab
GROUP BY GROUPING SETS (  
  (customer, year, product),
  (customer, year),
  (customer, product),
  (year, product),
  (customer),
  (year),
  (product));
```

<table>
<thead>
<tr>
<th>CUSTOMER</th>
<th>YEAR</th>
<th>PRODUCT</th>
<th>SUM(SALES)</th>
<th>GROUPING_ID(CUSTOMER,YEAR,PRODUCT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>2009</td>
<td>P1</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>C1</td>
<td>2010</td>
<td>P1</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>C2</td>
<td>2009</td>
<td>P1</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>C2</td>
<td>2010</td>
<td>P1</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>C1</td>
<td>2009</td>
<td>P2</td>
<td>200</td>
<td>0</td>
</tr>
<tr>
<td>C1</td>
<td>2010</td>
<td>P2</td>
<td>150</td>
<td>0</td>
</tr>
<tr>
<td>C2</td>
<td>2009</td>
<td>P2</td>
<td>300</td>
<td>0</td>
</tr>
<tr>
<td>C2</td>
<td>2010</td>
<td>P2</td>
<td>150</td>
<td>0</td>
</tr>
<tr>
<td>C1</td>
<td>2009</td>
<td>a</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>C1</td>
<td>2010</td>
<td>a</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>2009</td>
<td>a</td>
<td>500</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>2010</td>
<td>a</td>
<td>250</td>
<td>1</td>
</tr>
<tr>
<td>C1</td>
<td>a</td>
<td>P1</td>
<td>150</td>
<td>2</td>
</tr>
<tr>
<td>C2</td>
<td>a</td>
<td>P1</td>
<td>300</td>
<td>2</td>
</tr>
<tr>
<td>C1</td>
<td>a</td>
<td>P2</td>
<td>350</td>
<td>2</td>
</tr>
<tr>
<td>C2</td>
<td>a</td>
<td>P2</td>
<td>450</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>2009</td>
<td>P1</td>
<td>300</td>
<td>4</td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>YEAR</td>
<td>PRODUCT</td>
<td>SUM(SALES)</td>
<td>GROUPING_ID(CUSTOMER,YEAR,PRODUCT)</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>---------</td>
<td>------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>a</td>
<td>2010</td>
<td>P1</td>
<td>150</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>2009</td>
<td>P2</td>
<td>500</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>2010</td>
<td>P2</td>
<td>300</td>
<td>4</td>
</tr>
<tr>
<td>C1</td>
<td>a</td>
<td>a</td>
<td>500</td>
<td>3</td>
</tr>
<tr>
<td>C2</td>
<td>a</td>
<td>a</td>
<td>750</td>
<td>3</td>
</tr>
<tr>
<td>a</td>
<td>2009</td>
<td>a</td>
<td>800</td>
<td>5</td>
</tr>
<tr>
<td>a</td>
<td>2010</td>
<td>a</td>
<td>450</td>
<td>5</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>P1</td>
<td>450</td>
<td>6</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>P2</td>
<td>800</td>
<td>6</td>
</tr>
</tbody>
</table>

### 4.9.1.90 HAMMING_DISTANCE Function (String)

Performs a bitwise or bytewise comparison between two arguments and returns the hamming distance.

**Syntax**

```sql
HAMMING_DISTANCE(<left_hand_side>, <right_hand_side>)
```

**Syntax Elements**

- `left_hand_side`
  
  Specifies a left hand side argument against which `<right_hand_side>` is compared.
  
  ```sql
  <lhs> ::= <string>
  ```

- `right_hand_side`
  
  Specifies a right hand side argument to compare against `<left_hand_side>`.
  
  ```sql
  <rhs> ::= <string>
  ```
Description

Hamming distance describes differences between two specified arguments, and reflects the number of corresponding positions that differ from each other across the two arguments. Integer and binary arguments are compared bitwise, whereas string arguments are compared byte-wise.

The HAMMING_DISTANCE function returns -1 if the length of the two arguments is different, and NULL if either of the arguments is NULL.

Examples

The following example returns -1 because the arguments are of different length:

```
SELECT HAMMING_DISTANCE('abc', 'ca') "hamming_distance" FROM DUMMY;
```

The following example returns 0 because the arguments are identical:

```
SELECT HAMMING_DISTANCE('abc', 'abc') "hamming_distance" FROM DUMMY;
```

The following example returns 0 because the arguments are identical:

```
SELECT HAMMING_DISTANCE(4, 4) "hamming_distance" FROM DUMMY;
```

The following example returns 3 because all positions in the two arguments are different from the other string:

```
SELECT HAMMING_DISTANCE('abc', 'cab') "hamming_distance" FROM DUMMY;
```

The following example returns 4 to reflect the bitwise comparison of the arguments:

```
SELECT HAMMING_DISTANCE(TO_BINARY('abc'), TO_BINARY('cab')) "hamming_distance" FROM DUMMY;
```

The following example returns 3 to reflect the bitwise comparison of the arguments:

```
SELECT HAMMING_DISTANCE(4, 9) "hamming_distance" FROM DUMMY;
```

The following example returns -1 because the binary arguments are of different length:

```
SELECT HAMMING_DISTANCE(to_binary('abc'), to_binary('ca')) "hamming_distance" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]
4.9.1.91 HASH_MD5 Function (Miscellaneous)

Returns a 32 byte hash value of the concatenated arguments.

Syntax

```
HASH_MD5(<argument> [,...])
```

Syntax Elements

- **argument**
  Specifies a VARBINARY value.

```
<argument> ::= <string_literal>
```

Description

Returns a 32 byte VARBINARY hash value of the concatenated arguments. The hash is calculated using a MD5 algorithm.

To ensure unique results from concatenated arguments, delimit the arguments with another string as shown in following the examples.

Example

In the example below, the query generates a hash value based on the provided input:

```
SELECT HASH_MD5 (TO_BINARY('database')) "hash" FROM DUMMY;
```

The following examples, which both return 7AC66C0F148DE9519B8B264312C4D64, show how the function concatenates its arguments:

```
SELECT HASH_MD5(to_binary('abcd'), TO_BINARY('efg')) "test1" FROM DUMMY;
SELECT HASH_MD5(TO_BINARY('abc'), TO_BINARY('defg')) "test2" FROM DUMMY;
```

The following examples show how you can delimit the arguments with another string (in this case with 00) to ensure unique results from the concatenation. The first statement returns 6BB6BD45C57D57ECC69A9EB81F7409BE, whereas the second statement returns
Without the delimiter, both statements return the same value: 7AC66C0F148DE9519B8BD264312C4D64.

```sql
SELECT HASH_MD5(TO_BINARY('abcd'), '00', TO_BINARY('efg')) "test1" FROM DUMMY;
SELECT HASH_MD5(TO_BINARY('abc'), '00', TO_BINARY('defg')) "test2" FROM DUMMY;
```

### 4.9.1.92 HASH_SHA256 Function (Miscellaneous)

Returns a 32 byte hash value of the concatenated arguments.

**Syntax**

```
HASH_SHA256(<argument> [,....])
```

**Syntax Elements**

- **argument**
  - Specifies an argument of type VARBINARY.
  
  `<argument> ::= <string_literal>`

**Description**

Returns a 32 byte VARBINARY hash value of the concatenated arguments. The hash is calculated using a SHA256 algorithm.

To ensure unique results from concatenated arguments, delimit the arguments with another string as shown in following the examples.

**Example**

The following query returns a hash value for the specified arguments:

```sql
SELECT HASH_SHA256 (TO_BINARY('database')) "hash" FROM DUMMY;
```
The following examples, which both return 7D1A54127B222502F5B79B5FB080306152A44F92B37E23C6527BAF665D4DA9A, show how the function concatenates its arguments:

```
SELECT HASH_SHA256(TO_BINARY('abcd'), TO_BINARY('efg')) "test1" FROM DUMMY;
SELECT HASH_SHA256(TO_BINARY('abc'), TO_BINARY('defg')) "test2" FROM DUMMY;
```

The following examples show how you can delimit the arguments with another string (in this case with 00) to ensure unique results from the concatenation. The first statement returns 156D73B945474C6FB04D7CCAC1E31ACA9425F756801A0D87C3561FBE6661A659, whereas the second statement returns 6DE75B6290747BFA61C10FD0763344002C436475683B27E44C7C31FA920F1E3. Without the delimiter, both statements return the same value: 7D1A54127B222502F5B79B5FB080306152A44F92B37E23C6527BAF665D4DA9A.

```
SELECT HASH_SHA256(TO_BINARY('abcd'), '00', TO_BINARY('efg')) "test3" FROM DUMMY;
SELECT HASH_SHA256(TO_BINARY('abc'), '00', TO_BINARY('defg')) "test4" FROM DUMMY;
```

### 4.9.1.93 HEXTObIN Function (String)

Converts a string of hexadecimal characters to a VARBINARY value.

**Syntax**

```
HEXTOBIN(<hexadecimal_string>)
```

**Description**

HEXTOBIN returns a VARBINARY value where each byte of the result corresponds to two characters of `<hexadecimal_string>`. If `<hexadecimal_string>` does not contain an even number of digits, an error is returned.

In SAP HANA, lowercase characters in `<hexadecimal_string>` are supported and are treated as uppercase.

**Example**

The following two examples return the VARBINARY value 608DA975:

```
SELECT HEXTOBIN ('608da975') "Result" FROM DUMMY;
```
Related Information

Character String Data Types [page 36]

4.9.1.94 HEXTONUM Function (String)

Converts a hexadecimal value to a BIGINT string value.

Syntax

HEXTONUM(<string> [, -1])

Syntax Elements

- **string** Specifies the hexadecimal value to be converted to a string value.
- **-1**

The result is interpreted as a negative number if a second parameter is provided.

Description

Converts a hexadecimal value to a string value as a BIGINT data type. The input value can be a VARCHAR or NVARCHAR string type that is no longer than 16 characters.

Example

The following example converts the hexadecimal value `c2` to a BIGINT string value 12,345:

```sql
CREATE TABLE t2 (c2 VARCHAR(16));
INSERT INTO t2 VALUES('7FFFFFFFFFFFFFFFFFFF');
INSERT INTO t2 VALUES('8000000000000000');
INSERT INTO t2 VALUES ('FFFFFFFFFFFFFFF');
INSERT INTO t2 VALUES ('0');
INSERT INTO t2 VALUES ('3039');
```
SELECT c2, HEXTONUM(c2) FROM t2;

Related Information

Character String Data Types [page 36]

4.9.1.95 HOUR Function (Datetime)

Returns an integer representation of the hour portion of the specified time.

Syntax

HOUR(<time>)

Description

Returns an integer representation of the hour portion of the specified time.

Example

The following example returns the hour 12:

```
SELECT HOUR ('12:34:56') "hour" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]
4.9.1.96 IFNULL Function (Miscellaneous)

Returns the first non-NULL input expression.

Syntax

\[
\text{IFNULL}(\text{expression1}, \text{expression2})
\]

Description

IFNULL returns the first non-NULL input expression. If the data types of \text{expression1} and \text{expression2} are different, then SAP HANA chooses the data type with the higher precedence. For example, between TIMESTAMP and STRING, TIMESTAMP has the higher precedence. See the data types documentation for more information about data type precedence.

- Returns \text{expression1} if \text{expression1} is not NULL.
- Returns \text{expression2} if \text{expression1} is NULL.
- Returns NULL if both input expressions are NULL.

Example

The following query returns \text{diff}:

\[
\text{SELECT IFNULL ('diff', 'same') "ifnull" FROM DUMMY;}
\]

The following query returns \text{same}:

\[
\text{SELECT IFNULL (NULL, 'same') "ifnull" FROM DUMMY;}
\]

The following query returns NULL:

\[
\text{SELECT IFNULL (NULL, NULL) "ifnull" FROM DUMMY;}
\]

Related Information

Expressions [page 56]
Data Types [page 33]
4.9.1.97 INDEXING_ERROR_CODE Function (Fulltext)

Returns the indexing error codes for values in a column that has a fulltext index on it.

Syntax

```
INDEXING_ERROR_CODE(<column_name>)
```

Syntax Elements

- `<column_name>`
  Specifies the column to return the indexing error codes for.

```
<column_name> ::= <identifier>
```

Description

You must have a fulltext index on `<column_name>`.

Examples

The following statements create a table, build a fulltext index on it, insert some data, and then return the indexing error codes for the entries in the table.

```
DROP TABLE T;
CREATE COLUMN TABLE T (CONTENT varchar(50));
CREATE FULLTEXT INDEX I ON T(CONTENT) ASYNC MIME TYPE 'text/xml' FAST PREPROCESS OFF;
INSERT INTO T VALUES ('This is an example');
INSERT INTO T VALUES ('<xml>This is an example</xml>');
-- Wait a few seconds
SELECT INDEXING_ERROR_CODE(CONTENT),CONTENT FROM T;
```

<table>
<thead>
<tr>
<th>INDEXING_ERROR_CODE(CONTENT)</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5122</td>
<td>This is an example.</td>
</tr>
<tr>
<td>0</td>
<td>&lt;xml&gt;This is an example.&lt;/xml&gt;</td>
</tr>
</tbody>
</table>
4.9.1.98 INDEXING_ERROR_MESSAGE Function (Fulltext)

Returns the indexing error messages for values in a column that has a fulltext index on it.

**Syntax**

INDEXING_ERROR_MESSAGE(<column_name>)

**Syntax Elements**

- **column_name**
  Specifies the column to return indexing error messages from.

  `<column_name> ::= <identifier>`

**Description**

The INDEXING_ERROR_MESSAGE function returns the indexing error message for entries in `<column_name>`. You must have a fulltext index on `<column_name>`.

**Examples**

The following statements create a table, create a fulltext index on the table, insert data into the table, and then return the indexing errors for the entries:

```sql
CREATE COLUMN TABLE T (CONTENT VARCHAR(50));
CREATE FULLTEXT INDEX I ON T(CONTENT) ASYNC MIME TYPE 'text/xml' FAST PREPROCESS OFF;
INSERT INTO T VALUES ('This is an example');
INSERT INTO T VALUES ('<xml>This is an example</xml>');
SELECT INDEXING_ERROR_MESSAGE(CONTENT) "error message", CONTENT FROM T;
```

<table>
<thead>
<tr>
<th>error message</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xerces parser error</td>
<td>This is an example.</td>
</tr>
<tr>
<td>No error</td>
<td>&lt;xml&gt;This is an example.&lt;/xml&gt;</td>
</tr>
</tbody>
</table>
4.9.1.99 INDEXING_STATUS Function (Fulltext)

Returns the indexing status of values in a column that has a fulltext index.

Syntax

```
INDEXING_STATUS(<column_name>)
```

Description

This function returns the indexing status of values in a column that has a fulltext index. Possible return values are:

- ERROR - the value could not be indexed due to a mismatch in the data type
- INDEXED - the value is been indexed
- QUEUED - the value is queued for indexing

You must have a fulltext index on `<column_name>` to use this function.

Examples

```sql
DROP TABLE T;
CREATE TABLE T (CONTENT VARCHAR(500));
CREATE FULLTEXT INDEX I ON T(CONTENT) ASYNC MIME TYPE 'text/xml' FAST PREPROCESS OFF;
INSERT INTO T VALUES ('This string cannot be indexed due to its formatting, so its INDEXING_STATUS is ERROR');
INSERT INTO T VALUES ('<xml>This string was indexed, so its INDEXING_STATUS is INDEXED</xm>');
-- Wait a couple of seconds
ALTER FULLTEXT INDEX I SUSPEND QUEUE;
INSERT INTO T VALUES ('<xml>This string is queued for indexing, so its INDEXING_STATUS is QUEUED</xm>');
SELECT INDEXING_STATUS(CONTENT), CONTENT FROM T;
```

<table>
<thead>
<tr>
<th>INDEXING_STATUS(CONTENT)</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>This string cannot be indexed due to its formatting, so its INDEXING_STATUS is ERROR</td>
</tr>
<tr>
<td>INDEXED</td>
<td>&lt;xml&gt;This string was indexed, so its INDEXING_STATUS is INDEXED&lt;/xml&gt;</td>
</tr>
</tbody>
</table>
4.9.1.100 INITCAP Function (String)

Converts the first character of each word in a specified string to uppercase and converts remaining characters to lowercase.

Syntax

```sql
INITCAP(<inputString>)
```

Syntax Elements

- **inputString**
  Specifies the string to be converted. `<inputString>` can be either a VARCHAR or NVARCHAR value.

Description

A word is delimited by any of the following characters:

- Blank space
- New line
- Form feed
- Carriage return
- Line feed
- Any of the following: `! " # $ % & ' ( ) * + , - . / : ; < = > ? [ \ ] ^ _ ` { | } ~`
Example

The following query returns 'The Example One':

```sql
SELECT INITCAP('the EXAMPLE one') FROM DUMMY;
```

Related Information

Character String Data Types [page 36]

4.9.1.101 IS_SQL_INJECTION_SAFE Function (Security)

Checks a specified SQL identifier for possible SQL injection risks.

Syntax

```sql
IS_SQL_INJECTION_SAFE(<value>[, <max_tokens>])
```

Syntax Elements

- **value**
  Specifies the string to be checked.

  ```sql
  <value> ::= <string>
  ```

- **max_tokens**
  Specifies the maximum number of tokens expected in `<value>`. The default value is 1.

  ```sql
  <max_tokens> ::= <integer>
  ```

Description

Use this function to ensure a string contains the specified number of tokens and SQL comments before using it in an SQL statement. A 1 (safe) is returned when the actual number of tokens found does not exceed `<max_tokens>` and no comments were found; otherwise, a 0 is returned (not safe).
For the purposes of counting, SAP HANA interprets whitespaces and the following characters as separators between tokens:

, ( ) [ ] . : ; + - * / % ^ < > =

Example

The following query returns 0 (not safe) because two tokens were found and the default to check for was 1.

```sql
SELECT IS_SQL_INJECTION_SAFE('tab,le') "safe" FROM DUMMY;
```

The following query returns 1 (safe) because the number of tokens found did not exceed `<max_tokens>`:

```sql
SELECT IS_SQL_INJECTION_SAFE('CREATE STRUCTURED PRIVILEGE', 3) "safe" FROM DUMMY;
```

The following query returns 0 (not safe) because comments are not allowed.

```sql
SELECT IS_SQL_INJECTION_SAFE('mytab /*', 4) "safe" FROM DUMMY;
```

4.9.1.102 ISOWEEK Function (Datetime)

Returns the ISO year and week number for a specified date.

Syntax

```sql
ISOWEEK(<date>)
```

Description

Returns the ISO year and week numbers of the date specified by `<date>`. The week number is prefixed by the letter W.

Both the WEEK and ISOWEEK functions return the week number for a specified date but the format of the result is quite different, and the two functions may handle the first week of the new year differently. For example, when supplied the date 2017-01-01, the WEEK function considers the date to be part of the first week of 2017 and returns 1, whereas ISOWEEK considers the date to be part of the last week of 2016 and returns 2016-W52.

ISOWEEK has either 52 or 53 full weeks, with the extra week considered to be a leap week.
Example

The following example returns the value 2011-W22 for the ISO year and week numbers of the specified date:

```
SELECT ISOWEEK (TO_DATE('2011-05-30', 'YYYY-MM-DD')) "isoweek" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]

4.9.1.103 JSON_QUERY Function (JSON)

Extracts JSON text from a JSON context item by using a SQL/JSON path expression.

Syntax

```
JSON_QUERY (
  <JSON_API_common_syntax>
  [ <JSON_output_clause> ]
  [ <JSON_query_wrapper_behavior> ]
  [ <JSON_query_empty_behavior> ON EMPTY ]
  [ <JSON_query_error_behavior> ON ERROR ]
)
```

Syntax Elements

**JSON_API_common_syntax**

Specifies a JSON context item and a path to the context item, using common syntax for the JSON API.

```
<JSON_API_common_syntax> ::= <JSON_context_item>, <JSON_path_specification>
```

**JSON_context_item**

Specifies the JSON document to operate on, such as a table column, string, or collection.

**JSON_path_specification**

Specifies the path to `<JSON_context_item>`.

```
<JSON_path_specification> ::= <JSON_path_mode> <JSON_path_wff>
<JSON_path_mode> ::= STRICT | LAX
```
If a structural error occurs within a JSON filter expression and `<JSON_path_mode>` is set to STRICT, then the error handling of a JSON filter expression applies. Otherwise, a structural error is an unhandled error.

When the path is set to LAX, one of the following options occurs:

- If an operation requires an SQL/JSON array but the operand is not an SQL/JSON array, then the operand is wrapped in an SQL/JSON array prior to performing the operation.
- If an operation requires something other than an SQL/JSON array, but the operand is an SQL/JSON array, then the operand is unwrapped by converting its elements into an SQL/JSON sequence prior to performing the operation.

Array indexes start from 0, rather than 1 (the SQL standard).

If there is still a structural error after applying these resolutions, then the result is an empty SQL/JSON sequence.

`<JSON_path_wff>` indicates an actual JSON path (for example, `$item1`).
`<JSON_path_specification>` does not use double quotes.

**JSON_output_clause**

Specifies the output created by the JSON_QUERY function.

```
<JSON_output_clause> ::= RETURNING <data_type>
```

**data_type**

Specifies the data type to be set as the return type of the JSON_QUERY function. Supported data types: NVARCHAR(<length>), VARCHAR(<length>).

**JSON_query_wrapper_behavior WRAPPER**

Specifies the wrapper behavior of the JSON query.

```
<JSON_query_wrapper_behavior> WRAPPER
<JSON_query_wrapper_behavior> ::= WITHOUT [ ARRAY ]
    | WITH [ CONDITIONAL | UNCONDITIONAL ] [ ARRAY ]
```

The default is WITHOUT ARRAY WRAPPER. However, if WITH is specified, then the default is UNCONDITIONAL ARRAY WRAPPER.

**JSON_query_empty_behavior ON EMPTY**

Specifies the behavior of the function if the related data is not in the context item. The default is NULL ON EMPTY.

```
<JSON_query_empty_behavior> ON EMPTY
<JSON_query_empty_behavior> ::= ERROR
    | NULL
    | EMPTY ARRAY
    | EMPTY OBJECT
```

ERROR ON EMPTY returns an error if the related data is not in the context item. NULL ON EMPTY returns a NULL if the related data is not in the context item. EMPTY ARRAY ON EMPTY returns an empty array if the related data is not in the context item. EMPTY OBJECT ON EMPTY returns an empty JSON object if the related data is not in the context item.

**JSON_query_error_behavior ON ERROR**
Specifies the behavior of the function when the query throws an error. The default is NULL ON ERROR.

```sql
<JSON_query_error_behavior> ON ERROR
<JSON_query_error_behavior> ::= ERROR | NULL | EMPTY ARRAY | EMPTY OBJECT
```

ERROR ON ERROR returns an error if the function result includes an error. NULL ON ERROR returns a NULL if the function result includes an error. EMPTY ARRAY ON ERROR returns an empty array if the function result includes an error. EMPTY OBJECT ON ERROR returns an empty JSON object if the function result includes an error.

**Description**

Extracts JSON text from a JSON context item using a SQL/JSON path expression.

The following tokens are supported in `<JSON_API_common_syntax>`:

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>The context item (the first argument of the function).</td>
<td>'$'</td>
</tr>
<tr>
<td>.</td>
<td>The member of an object.</td>
<td>`.item.description'</td>
</tr>
<tr>
<td>[</td>
<td>The array index specifier (open).</td>
<td>'[$[1]'</td>
</tr>
<tr>
<td>]</td>
<td>The array index specifier (closed).</td>
<td>'[$[1]'</td>
</tr>
<tr>
<td>to</td>
<td>The array index range.</td>
<td>'$[3 to 5]'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= '[$[3,4,5]'</td>
</tr>
<tr>
<td>*</td>
<td>The wild card.</td>
<td>'.*.description'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'.<em>.item.list[</em>]'</td>
</tr>
</tbody>
</table>

**Example**

The following query returns the value `{"item1":1,"item2":2,"item3":3}`.

```sql
SELECT JSON_QUERY('{"item1":1, "item2":2, "item3":3}', '$') AS JSONQUERY FROM DUMMY;
```
The following query returns the value [1].
SELECT JSON_QUERY('{"item1":1, "item2":2, "item3":3}', '$.item1' WITH WRAPPER )
AS JSONQUERY FROM DUMMY;

Related Information
Expressions [page 56]

4.9.1.104 JSON_TABLE Function (JSON)
Queries a JSON text and presents it as a relational table.

Syntax
JSON_TABLE(
<JSON_API_common_syntax>
<JSON_table_columns_clause>
[ <JSON_table_error_behavior> ON ERROR ]
)

Syntax Elements
JSON_API_common_syntax
Specifies a JSON context item and a path to the context item, using common syntax for the JSON API.
<JSON_API_common_syntax> ::= <JSON_context_item>, <JSON_path_specification>
JSON_context_item
Specifies the JSON document to operate on, such as a table column, string, or collection.
JSON_path_specification
Specifies the path to <JSON_context_item>.
<JSON_path_specification> ::= <JSON_path_mode> <JSON_path_wff>
<JSON_path_mode> ::= STRICT | LAX
Depending on the location in the grammar, there are three kinds of JSON path expressions:
Row pattern path expression Used to produce an SQL/JSON sequence, with one SQL/JSON item
for each row of the output table.

SAP HANA SQL Reference Guide for SAP HANA Platform
SQL Reference

PUBLIC

223


**Column pattern path expression** Used to search for the column within the current SQL/JSON item produced by the row pattern.

**Nested columns pattern path expression** Used for unnesting of (even deeply) nested JSON objects/arrays in one invocation rather than chaining several JSON_TABLE expressions in the SQL-statement.

For example:

<table>
<thead>
<tr>
<th>Path Expression Type</th>
<th>Example Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row pattern path expression</td>
<td>( \text{lax } $ )</td>
</tr>
<tr>
<td>Column pattern path expression</td>
<td>( \text{lax } $.name )</td>
</tr>
<tr>
<td></td>
<td>( \text{lax } $.type )</td>
</tr>
<tr>
<td></td>
<td>( \text{lax } $.number )</td>
</tr>
<tr>
<td>Nested pattern path expression</td>
<td>( \text{lax } $.phoneNumber[*] )</td>
</tr>
</tbody>
</table>

```sql
SELECT bookclub.id, jt.name, jt.type, jt.number FROM bookclub,
   JSON_TABLE ( bookclub.jcol, 'lax $'
   COLUMNS ( name VARCHAR(30) PATH 'lax $.Name',
      NESTED PATH 'lax $.phoneNumber[*]' COLUMNS ( type VARCHAR(10) PATH 'lax $.type',
      number CHAR(12) PATH 'lax $.number' )
   ) AS jt;
```

The error types generated by the `<JSON_path_specification>` are:

- an input conversion error (for example, invalid JSON document – cannot be parsed)
- an error returned by the PATH engine, which evaluates JSON path expressions
- a structural error meaning no matching path in the JSON document (for example, \'$.ab\' path expression against {"cd": 1})
  - If a structural error occurs within a JSON filter expression and `<JSON_path_mode>` is set to STRICT, then the error handling of a JSON filter expression applies. Otherwise, a structural error is an unhandled error.
  - When the path is set to LAX, one of the following options occurs:
    - If an operation requires an SQL/JSON array but the operand is not an SQL/JSON array, then the operand is wrapped in an SQL/JSON array prior to performing the operation.
    - If an operation requires something other than an SQL/JSON array, but the operand is an SQL/JSON array, then the operand is unwrapped by converting its elements into an SQL/JSON sequence prior to performing the operation.
  - Array indexes start from 0, rather than 1 (the SQL standard).
  - If there is still a structural error after applying these resolutions, then the result is an empty SQL/JSON sequence.

`<JSON_path_wff>` indicates an actual JSON path (for example, \'$.item1\').
`<JSON_path_specification>` does not use double quotes.

**JSON_table_columns_clause**
Specifies the columns that are created.

```sql
<JSON_table_columns_clause> ::= 
  COLUMNS ( <JSON_table_column_definition> [,... ] )
```

**JSON_table_column_definition**

Defines the columns generated.

**JSON_table_ordinality_column_definition**

Defines an ordinality column. An ordinality column is similar to a column defined using the `ROW_NUMBER` window function.

```sql
<JSON_table_ordinality_column_definition> ::= <column_name> FOR ORDINALITY
```

**JSON_table_regular_column_definition**

Defines the regular columns. Each result (row) of a regular JSON table column is equivalent to a `JSON_VALUE` function result.

```sql
<JSON_table_regular_column_definition> ::= <column_name> <data_type> 
  PATH <JSON_table_column_path_specification> 
  [ <JSON_table_column_empty_behavior> ON EMPTY ] 
  [ <JSON_table_column_error_behavior> ON ERROR ] 
<JSON_table_column_path_specification> ::= <JSON_path_specification>
```

**column_name**

Specifies the name of the column.

**data_type**

Specifies the data type.

```sql
<data_type> ::=  BIGINT 
  DATE 
  DECIMAL 
  DOUBLE 
  INT 
  NVARCHAR (<int_const>) 
  SECONDDATE 
  SMALLDECIMAL 
  TIME 
  TIMESTAMP 
  VARCHAR (<int_const>)
```

**JSON_table_column_path_specification**
Specifies the JSON path that specifies which JSON value the JSON context item is extracted from.

**table_column_empty_behavior ON EMPTY**

Specifies the behavior of the function when the created column is empty. The default is NULL ON EMPTY.

ERROR ON EMPTY returns an error when the created column is empty. NULL ON EMPTY returns a NULL when the created column is empty. DEFAULT `<value_expression>` ON EMPTY returns `<value_expression>` when the created column is empty.

**JSON_table_column_error_behavior ON ERROR**

Specifies the behavior of the function when there is an error during column creation. The default is NULL ON ERROR.

ERROR ON ERROR results in an error being thrown if the function result includes an error. NULL ON ERROR returns a NULL if the function result includes an error. DEFAULT `<value_expression>` ON ERROR returns `<value_expression>` if the function result includes an error.

**JSON_table_formatted_column_definition**

Specifies the column definition for formatted columns where the records in the column are formatted in JSON syntax. Each result (row) of a regular JSON table column is equivalent to a JSON_QUERY function result.

```sql
<JSON_table_formatted_column_definition> ::=    <column_name> <data_type>   FORMAT <JSON_representation>   PATH <JSON_table_column_path_specification>   [ <JSON_table_formatted_column_empty_behavior> ON EMPTY ]   [ <JSON_table_formatted_column_error_behavior> ON ERROR ]
```

**column_name**

Specifies the name of the column.

**data_type**

Specifies the data type to be set as the return type of the function. Supported data types for formatted columns are VARCHAR(<n>) and NVARCHAR(<n>).

**JSON_representation**

Specifies the JSON encoding to use.

```sql
<JSON_representation> ::=    JSON    JSON ENCODING { UTF8 }    JSON ENCODING { UTF16 }    JSON ENCODING { UTF32 }
```

**JSON_table_column_path_specification**

Specifies the JSON path that specifies which JSON value the JSON context item is extracted from.

**JSON_table_formatted_column_wrapper_behavior WRAPPER**

Specifies the wrapper behavior of the formatted column.

```sql
<JSON_table_formatted_column_wrapper_behavior> ::=    WITHOUT [ ARRAY ]    | WITH [ CONDITIONAL | UNCONDITIONAL ] [ ARRAY ]
```
When WITHOUT [ ARRAY ] WRAPPER is specified, the formatted column is not represented as a JSON array. When WITH [ CONDITIONAL | UNCONDITIONAL ] [ ARRAY ] WRAPPER is specified, the formatted column is set with a conditional/unconditional array. With a conditional array, the result is formatted as a JSON array if the result is neither a JSON array nor a JSON object. With an unconditional array, the result is formatted as a JSON array if the result is not a JSON object. The difference between unconditional and conditional wrapper is that unconditional wrapper wraps the result as a JSON array once more when the result is a JSON array.

**JSON_table_formatted_column_empty_behavior ON EMPTY**

Specifies the JSON behavior if the related data is not in the context item.

```
<JSON_table_formatted_column_empty_behavior> ::= ERROR | NULL | EMPTY ARRAY | EMPTY OBJECT
```

ERROR ON EMPTY returns an error if the related data is not in the context item. NULL ON EMPTY returns a NULL if the related data is not in the context item. EMPTY ARRAY ON EMPTY returns an empty array if the related data is not in the context item. EMPTY OBJECT ON EMPTY returns an empty JSON object if the related data is not in the context item.

**JSON_table_formatted_column_error_behavior ON ERROR**

Specifies the behavior when the formatted column throws an error.

```
<JSON_table_formatted_column_error_behavior> ::= ERROR | NULL | EMPTY ARRAY | EMPTY OBJECT
```

ERROR ON ERROR returns an error if the function returns no result. NULL ON ERROR returns a NULL if the function returns no result. EMPTY ARRAY ON ERROR returns an empty array if the result of the formatted column is empty. EMPTY OBJECT ON ERROR returns an empty JSON object if the result of the formatted column is empty.

**JSON_table_nested_columns**

Defines nested columns.

```
<JSON_table_nested_columns> ::= NESTED [ PATH ]
<JSON_table_nested_path_specification>
<JSON_table_columns_clause>
<JSON_table_nested_path_specification> ::= <JSON_path_specification>

• <JSON_table_columns_clause>: Specifies the columns that are created with the function.
• <JSON_path_specification>: The JSON path which specifies which JSON value from the JSON context item is extracted.
```

**JSON_table_error_behavior**

Specifies the behavior of the function when an error occurs. The default behavior is EMPTY ON ERROR.

```
<JSON_table_error_behavior> ::= { ERROR | EMPTY }
```

ERROR

If the function result includes an error, then the error is returned.
If the function result includes an error, then it returns an empty result.

**Description**

Queries a JSON text and presents it as a relational table.

The following tokens are supported in `<JSON_API_common_syntax>`:

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>The current context item.</td>
<td><code>'$'</code></td>
</tr>
<tr>
<td>.</td>
<td>The member of an object.</td>
<td><code>$.item.description</code></td>
</tr>
<tr>
<td>[</td>
<td>The array index specifier (open).</td>
<td></td>
</tr>
<tr>
<td>]</td>
<td>The array index specifier (closed).</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>The array index range.</td>
<td><code>'$[3 to 5]'</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>= <code>'$[3,4,5]'</code></td>
</tr>
<tr>
<td>*</td>
<td>The wild card.</td>
<td><code>$.item.list[*]'</code></td>
</tr>
</tbody>
</table>

**Example**

The following examples use the table created below:

```sql
CREATE ROW TABLE T1 (A INT, B NVARCHAR(5000));
INSERT INTO T1 VALUES (1, '"
'       "PONumber": 1,
       "Reference": "BSMITH-74635645",
       "Requestor": "Barb Smith",
       "User": "BSMITH",
       "CostCenter": "A50",
       "ShippingInstructions":
           {
           "name": "Barb Smith",
           "Address":
               {
               "street": "100 Fairchild Ave",
               "city": "San Diego",
               "state": "CA",
               "zipCode": 23345,
               "country": "USA"
               },
```
The following example selects from an ordinality column and a regular column:

```sql
SELECT JT.*
FROM JSON_TABLE(T1.B, '$.LineItems[*]' 
  COLUMNS
    ( 
      RN FOR ORDINALITY,
      ITEM_NUMBER INT PATH '$.ItemNumber',
      UPC_CODE BIGINT PATH '$.Part.UPCCode'
    ) ) AS JT;
```

<table>
<thead>
<tr>
<th>RN</th>
<th>ITEM_NUMBER</th>
<th>UPC_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>73649587162</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>83600229374</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>33298003521</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>91827739856</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>22983303876</td>
</tr>
</tbody>
</table>

The following example selects from a formatted column:

```sql
SELECT * 
FROM JSON_TABLE(T1.B, '$.ShippingInstructions' 
  COLUMNS
    ( 
      PHONE VARCHAR(50) FORMAT JSON PATH '$.Phone'
    ) ) AS JT;
```

PHONE

```json
["number":"519-555-6310","type":"Office"]
```

The following example selects from a nested column:

```sql
SELECT * 
FROM JSON_TABLE(T1.B, '$.ShippingInstructions'
```
COLUMNS
(
  NESTED PATH '$.Address'
  COLUMNS
  (
    STREET NVARCHAR(50) PATH '$.street',
    CITY NVARCHAR(50) PATH '$.city'
  )
) AS JT;

<table>
<thead>
<tr>
<th>STREET</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 Fairchild Ave</td>
<td>San Diego</td>
</tr>
</tbody>
</table>

The following example selects from an ordinality column with nested columns:

```sql
SELECT *
FROM JSON_TABLE(T1.B, '$'
  COLUMNS
  (RN FOR ORDINALITY,
    USER_NAME NVARCHAR(20) PATH '$.User',
    NESTED PATH '$.LineItems[1,2]'
  COLUMNS
    (ORDER_NUMBER FOR ORDINALITY,
      ITEM_NUMBER INT PATH '$.ItemNumber',
      QUANTITY INT PATH '$.Quantity'
    )
  )
) AS JT;
```

<table>
<thead>
<tr>
<th>RN</th>
<th>USER_NAME</th>
<th>ORDER_NUMBER</th>
<th>ITEM_NUMBER</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BSMITH</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>BSMITH</td>
<td>2</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>

The following example demonstrates the difference when specifying CONDITIONAL ARRAY WRAPPER versus UNCONDITIONAL ARRAY WRAPPER when returning a JSON array:

```sql
CREATE ROW TABLE r1 ( a INT, b NVARCHAR(5000));
INSERT INTO r1 VALUES (1, '{"menu": {"header": "SVG Viewer","items": [{"id": "Open"},{"id": "OpenNew", "label": "Open New"},null,{"id": "ZoomIn", "label": "Zoom In"}]}}');

SELECT JSON_QUERY(B, '$.menu.items' WITH UNCONDITIONAL ARRAY WRAPPER) FROM r1 WITH HINT(IGNORE_PLAN_CACHE);
```

```json
[{{"id":"Open"},{"id":"OpenNew","label":"Open New"},null,{{"id":"ZoomIn","label":"Zoom In"}}}]
```

```sql
SELECT JSON_QUERY(B, '$.menu.items' WITH CONDITIONAL ARRAY WRAPPER) FROM r1;
```

```json
[{{"id":"Open"},{"id":"OpenNew","label":"Open New"},null,{{"id":"ZoomIn","label":"Zoom In"}]}
```
The following example demonstrates an example of \texttt{ERROR ON ERROR}, which results in an error stating that the data for \texttt{"User"} cannot be parsed to an integer data type.

\begin{verbatim}
SELECT * FROM JSON_TABLE(T1.B, '$' COLUMNS (    RN FOR ORDINALITY,    USER_NAME INT PATH '$.User' ERROR ON ERROR ) ERROR ON ERROR ) AS JT;
\end{verbatim}

The following example demonstrates an example of \texttt{EMPTY ON ERROR}. The default behavior of \texttt{JSON_table_error_behavior} is \texttt{EMPTY ON ERROR}. It returns an error result instead of throwing empty when \texttt{JSON_table_column_error_behavior} is \texttt{ERROR ON ERROR} or \texttt{JSON_table_column_empty_behavior} is \texttt{ERROR ON EMPTY}.

\begin{verbatim}
SELECT * FROM JSON_TABLE(T1.B, '$' COLUMNS (    RN FOR ORDINALITY,    USER_NAME INT PATH '$.User' ERROR ON ERROR ) ) AS JT;
\end{verbatim}

**Related Information**

Expressions [page 56]

**4.9.1.105 JSON_VALUE Function (JSON)**

Extracts an SQL value of a predefined type from a JSON value.

**Syntax**

\begin{verbatim}
JSON_VALUE(   <JSON_API_common_syntax>   [ <JSON_returning_clause> ]   [ <JSON_value_empty_behavior> ON EMPTY ]   [ <JSON_value_error_behavior> ON ERROR ]
\end{verbatim}
Syntax Elements

**JSON_API_commonSyntax**

Specifies a JSON context item and a path to the context item, using common syntax for the JSON API.

```plaintext
<JSON_API_common_syntax> ::= <JSON_context_item>, <JSON_path_specification>
```

**JSON_context_item**

Specifies the JSON document to operate on, such as a table column, string, or collection.

**JSON_path_specification**

Specifies the path to <JSON_context_item>.

```plaintext
<JSON_path_specification> ::= <JSON_path_mode> <JSON_path_wff>
<JSON_path_mode> ::= STRICT | LAX
```

When `<JSON_path_mode>` is set to STRICT, if the structural error occurs within a JSON filter expression, then the error handling of a JSON filter expression applies. Otherwise, a structural error is an unhandled error.

When the path is set to LAX, one of the following options occurs:

- If an operation requires an SQL/JSON array but the operand is not an SQL/JSON array, then the operand is wrapped in an SQL/JSON array prior to performing the operation.
- If an operation requires something other than an SQL/JSON array, but the operand is an SQL/JSON array, then the operand is unwrapped by converting its elements into an SQL/JSON sequence prior to performing the operation.

Array indexes start from 0, rather than 1 (the SQL standard).

If there is still a structural error after applying these resolutions, then the result is an empty SQL/JSON sequence.

- `<JSON_path_wff>` indicates an actual JSON path (for example, `$.item1`).
- `<JSON_path_specification>` does not use double quotes.

**JSON_returning_clause**

Defines the data type of the result.

```plaintext
<JSON_returning_clause> ::= RETURNING <data_type>
<data_type> ::= INTEGER
| BIGINT
| DECIMAL
| VARCHAR(<integer>)
| NVARCHAR(<integer>)
```

**JSON_value_empty_behavior ON EMPTY**

232
Specifies the behavior of the function when the related data is not in the context item. The default is NULL ON EMPTY.

\[
<\text{JSON\_value\_empty\_behavior}> \text{ ON EMPTY} \\
<\text{JSON\_value\_empty\_behavior}> ::= \text{ERROR} \\
\quad | \text{NULL} \\
\quad | \text{DEFAULT } <\text{value\_expression}>
\]

ERROR ON EMPTY returns an error if the related data is not in the context item. NULL ON EMPTY returns a NULL if the related data is not in the context item. DEFAULT \(<\text{value\_expression}> \text{ ON EMPTY returns } <\text{value\_expression}> \) if the related data is not in the context item.

**JSON\_value\_error\_behavior ON ERROR**

Specifies the behavior of the JSON\_VALUE function when an error occurs. The default is NULL ON ERROR.

\[
<\text{JSON\_value\_error\_behavior}> \text{ ON ERROR} \\
<\text{JSON\_value\_error\_behavior}> ::= \text{ERROR} \\
\quad | \text{NULL} \\
\quad | \text{DEFAULT } <\text{value\_expression}>
\]

ERROR ON ERROR returns an error if the function results include an error. NULL ON ERROR returns a NULL if the function results include an error. DEFAULT \(<\text{value\_expression}> \text{ ON ERROR returns } <\text{value\_expression}> \) if the function results include an error.

**Description**

Extracts an SQL value of a predefined type from a JSON value.

The following tokens are supported in \(<\text{JSON\_API\_common\_syntax}>\):

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>The context item (the first argument of the function).</td>
<td>'$'</td>
</tr>
<tr>
<td>.</td>
<td>The member of an object.</td>
<td>'$.item.description'</td>
</tr>
<tr>
<td>[</td>
<td>The array index specifier (open).</td>
<td></td>
</tr>
<tr>
<td>]</td>
<td>Array index specifier (closed).</td>
<td>'$_[1]'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'$.item.list[1]'</td>
</tr>
<tr>
<td>to</td>
<td>The array index range.</td>
<td>'$_[3 \text{ to } 5]'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= '$_[3,4,5]'</td>
</tr>
<tr>
<td>*</td>
<td>The wild card.</td>
<td>'$_.*.description'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'$.item.list[*]'</td>
</tr>
</tbody>
</table>
Example
The following statement returns a value of 10:
SELECT JSON_VALUE('{"item1":10}', '$.item1') AS "value" FROM DUMMY;
The following statement returns a value of 5:
SELECT JSON_VALUE('{"item1":{"sub1":10}, "item2":{"sub2":5}, "item3":
{"sub3":7}}', '$.*.sub2') AS "value" FROM DUMMY;
The following statement returns a value of 0:
SELECT JSON_VALUE('[0, 1, 2, 3]', '$[0]') AS "value"

FROM DUMMY;

The following statement returns the value "No last name found":
SELECT JSON_VALUE('{"firstname":"John"}', '$.lastname' DEFAULT 'No last name
found' ON EMPTY) AS "Last Name" FROM DUMMY;
The following statement causes a type conversion error to demonstrate the behavior for ERROR ON ERROR:
SELECT JSON_VALUE('{"item":"string"}', '$.item' RETURNING DECIMAL ERROR ON
ERROR) AS "Item" FROM DUMMY;
The following statement demonstrates what happens when there is no value (the object does not have the
name "last name"):
SELECT JSON_VALUE('{"firstname":"John"}', 'strict $.lastname' ERROR ON ERROR) AS
"Last Name" FROM DUMMY;

Related Information
Expressions [page 56]

4.9.1.106 LAG Function (Window)
Returns the value of the offset rows before the current row.

Syntax

LAG( <expression> [, <offset> [, <default_expr> ] ] ) <window_specification>

234

PUBLIC

SAP HANA SQL Reference Guide for SAP HANA Platform
SQL Reference


Syntax Elements

offset

The `<offset>` should be non-negative and the default is 1.

If the `<offset>` crosses boundaries of the partition, then `<default_expr>` value is returned. If the `<default_expr>` is not specified, then a null value is returned. The `<offset>` and `<default_expr>` are evaluated at current row.

window_specification

Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].

Description

The output of the LAG function can be non-deterministic among tie values.

Examples

```
CREATE ROW TABLE T (class CHAR(10), val INT, offset INT);
INSERT INTO T VALUES('A', 1, 1);
INSERT INTO T VALUES('A', 3, 3);
INSERT INTO T VALUES('A', 5, null);
INSERT INTO T VALUES('A', 5, 2);
INSERT INTO T VALUES('A', 10, 0);
INSERT INTO T VALUES('B', 1, 3);
INSERT INTO T VALUES('B', 1, 1);
INSERT INTO T VALUES('B', 7, 1);

SELECT class,
    val,
    offset,
    LEAD(val) OVER (PARTITION BY class ORDER BY val) AS lead,
    LEAD(val, offset, -val) OVER (PARTITION BY class ORDER BY val) AS lead2,
    LAG(val) OVER (PARTITION BY class ORDER BY val) AS lag,
    LAG(val, offset, -val) OVER (PARTITION BY class ORDER BY val) AS lag2
FROM T;
```

<table>
<thead>
<tr>
<th>CLASS</th>
<th>VAL</th>
<th>OFFSET</th>
<th>LEAD</th>
<th>LEAD2</th>
<th>LAG</th>
<th>LAG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>null</td>
<td>-1</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>null</td>
<td>5</td>
<td>-5</td>
<td>3</td>
<td>-5</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>-5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>0</td>
<td>null</td>
<td>10</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>
4.9.1.107 LANGUAGE Function (Fulltext)

Returns the language of the specified column entries.

Syntax

```
LANGUAGE(<column_name>)
```

Description

`<column_name>` is the column where the language detection occurs.

You must have an existing fulltext index for the specified column.

Example

Create a table with two detectable languages, English and German, and populate it with some entries.

```
CREATE COLUMN TABLE T (CONTENT TEXT FAST PREPROCESS OFF LANGUAGE DETECTION('EN','DE'));
INSERT INTO T VALUES('This is a very short example.');
INSERT INTO T VALUES('Dies ist ein ganz kurzes Beispiel.');
```
Execute the following query to select the content and detect the language of the entries.

```
SELECT LANGUAGE(CONTENT), CONTENT FROM T;
```

The query returns the language of each column:

<table>
<thead>
<tr>
<th>LANGUAGE(CONTENT)</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>This is a very short example.</td>
</tr>
<tr>
<td>de</td>
<td>Dies ist ein ganz kurzes Beispiel.</td>
</tr>
</tbody>
</table>

### 4.9.1.108 LAST_DAY Function (Datetime)

Returns the date of the last day of the month that contains the specified date.

**Syntax**

```
LAST_DAY(<date>)
```

**Description**

Returns the date of the last day of the month that contains the date `<date>`.

**Example**

The following example returns the value 2010-01-31 (or another format like Jan 31, 2010, depending on your date display settings):

```
SELECT LAST_DAY (TO_DATE('2010-01-04', 'YYYY-MM-DD')) "last day" FROM DUMMY;
```

**Related Information**

Datetime Data Types [page 40]
4.9.1.109 LAST_VALUE Function (Aggregate)

Returns the value of the last element of an expression. This function can also be used as a window function.

Syntax

Aggregate function:

```
LAST_VALUE( <expression> <order_by_clause> )
```

Window function:

```
LAST_VALUE( <expression> <order_by_clause> ) <window_specification>
```

Syntax Elements

**expression**

Specifies the expression of data to operate over.

**order_by_clause**

Specifies the sort order of the input rows.

```
<order_by_clause> ::= ORDER BY <order_by_expression> [ , <order_by_expression> [... , ... ] ]
<order_by_expression> ::= <column_name> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [ <collate_clause> ]
| <column_position> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [ <collate_clause> ]
<collate_clause> ::= COLLATE <collation_name>
```

**<collate_clause>** specifies the collation to use for ordering values in the results. **<collate_clause>** can only be used on columns defined as NVARCHAR or VARCHAR. **<collation_name>** is one of the supported collation names listed in the COLLATIONS system view.

**window_specification**

Defines a window on the data over which the function operates. For **<window_specification>**, see Window Functions and the Window Specification [page 424].

Description

Returns the value of the last element in **<expression>** as ordered by **<order_byClause>**.

NULL is returned if the value is NULL or if **<expression>** is empty.

The output of the LAST_VALUE function can be non-deterministic among tie values.
Example

The example below returns the last value in COL1 column when the table is ordered by COL2. The query returns 4.

```
CREATE ROW TABLE T (COL1 DOUBLE, COL2 DOUBLE);
INSERT INTO T VALUES(1, 1);
INSERT INTO T VALUES(4, 5);
INSERT INTO T VALUES(7, 3);
SELECT LAST_VALUE (COL1 ORDER BY COL2) FROM T;
```

Example for Spatial Data Type (ST_Geometry, ST_Point)

The example below returns the last value, `POINT (4 4)`, in binary format, when the table is ordered by ID:

```
CREATE TABLE TAB (ID INT, SHAPE ST_Geometry(4326));
INSERT INTO TAB VALUES(1, ST_GeomFromText('POINT(1 1)', 4326));
INSERT INTO TAB VALUES(2, ST_GeomFromText('POINT(2 2)', 4326));
INSERT INTO TAB VALUES(3, ST_GeomFromText('POINT(3 3)', 4326));
INSERT INTO TAB VALUES(4, ST_GeomFromText('POINT(4 4)', 4326));
SELECT LAST_VALUE(SHAPE ORDER BY ID) FROM TAB;
```

To return a result as data type NVARCHAR, use the following statement:

```
SELECT LAST_VALUE(CAST(SHAPE AS NVARCHAR) ORDER BY ID) FROM TAB;
```

Related Information

- COLLATIONS System View [page 1520]
- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- Aggregate Functions [page 414]
- SAP HANA Spatial Reference

4.9.1.110 LCASE Function (String)

Converts all characters in a string to lowercase.

Syntax

```
LCASE(<string>)
```
**Description**

Converts all characters in string `<string>` to lowercase. The LCASE function is identical to the LOWER function.

**Example**

This example converts all characters of the given string TesT to lowercase and returns the value test.

```sql
SELECT LCASE ('TesT') "lcase" FROM DUMMY;
```

**Related Information**

LOWER Function (String) [page 255]
Character String Data Types [page 36]

---

**4.9.1.111 LEAD Function (Window)**

Returns the offset of rows after the current row. The `<offset>` should be non-negative and its default is 1.

**Syntax**

```
LEAD( <expression> [, <offset> [, <default_expr> ] ] ) <window_specification>
```

**Syntax Elements**

**offset**

If the `<offset>` crosses boundaries of the partition, then the `<default_expr>` value is returned. If the `<default_expr>` is not specified, then a null value is returned. The `<offset>` and `<default_expr>` are evaluated at current row.

**window_specification**

Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].
**Description**

The output of the LEAD function can be non-deterministic among tie values.

**Examples**

```sql
CREATE ROW TABLE T (class CHAR(10), val INT, offset INT);
INSERT INTO T VALUES('A', 1, 1);
INSERT INTO T VALUES('A', 3, 3);
INSERT INTO T VALUES('A', 5, null);
INSERT INTO T VALUES('A', 5, 2);
INSERT INTO T VALUES('A', 10, 0);
INSERT INTO T VALUES('B', 1, 3);
INSERT INTO T VALUES('B', 1, 1);
SELECT class,
    val,
    offset,
    LEAD(val) OVER (PARTITION BY class ORDER BY val) AS lead,
    LEAD(val, offset, -val) OVER (PARTITION BY class ORDER BY val) AS lead2,
    LAG(val) OVER (PARTITION BY class ORDER BY val) AS lag,
    LAG(val, offset, -val) OVER (PARTITION BY class ORDER BY val) AS lag2
FROM T;
```

<table>
<thead>
<tr>
<th>CLASS</th>
<th>VAL</th>
<th>OFFSET</th>
<th>LEAD</th>
<th>LEAD2</th>
<th>LAG</th>
<th>LAG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>null</td>
<td>-1</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>null</td>
<td>5</td>
<td>-5</td>
<td>3</td>
<td>-5</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>2</td>
<td>10</td>
<td>-5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>0</td>
<td>null</td>
<td>10</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>null</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>1</td>
<td>null</td>
<td>-7</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Related Information**

Expressions [page 56]
Window Functions and the Window Specification [page 424]
4.9.1.112  LEAST Function (Miscellaneous)

Returns the lesser value of two specified arguments.

Syntax

```
LEAST(<argument_1> [, <argument_2>]...)
```

Description

Returns the lesser value of the arguments. If one of the arguments is NULL, the return is NULL.

Example

The following query returns aa.

```
SELECT LEAST('aa', 'ab', 'ba', 'bb') "least" FROM DUMMY;
```

4.9.1.113  LEFT Function (String)

Returns the specified number of characters or bytes of a string, starting from the left side.

Syntax

```
LEFT(<string>, <number>)
```

Syntax Elements

- **string**
  - Specifies the string to be operated on.
- **number**
  - Specifies the length of characters or bytes to return, starting from the left side.
**Description**

Returns the first `<number>` of characters or bytes from the beginning of `<string>`.

Returns an empty string value if `<number>` is less than 1.

Returns `<string>` (without blank padding) if the value of `<number>` is greater than the length of `<string>`.

**Example**

The following example returns the leftmost three characters of the string **Hel**:

```sql
SELECT LEFT ('Hello', 3) "left" FROM DUMMY;
```

The following example returns the string **Hello** because the value 10 exceeds the string length:

```sql
SELECT LEFT ('Hello', 10) "left" FROM DUMMY;
```

**4.9.1.114  LENGTH Function (String)**

Returns the number of characters in a string.

**Syntax**

```
LENGTH(<string>)
```

**Description**

Returns the number of characters in string `<string>`.

If `<string>` is a VARCHAR, then the function does not return the number of bytes. In this case, the number of characters like NVARCHAR-typed strings are returned instead.

Supplementary plane Unicode characters, each of which occupies 6 bytes in CESU-8 encoding, are counted as two characters.
**Example**

This example returns the number of characters (14) contained in the given string:

```sql
SELECT LENGTH ('length in char') "length" FROM DUMMY;
```

**Related Information**

[Character String Data Types](#) [page 36]

### 4.9.1.115 LINEAR_APPROX Function (Window)

Operates on an entire series to produce a new series that replaces missing values by interpolating between adjacent non-NULL values and extrapolating any leading or trailing null values.

**Syntax**

```sql
LINEAR_APPROX(<expression> [, <ModeArgument> [, <Value1Argument> [, <Value2Argument>]]]) OVER ({ SERIES TABLE <table_schema> 
    [<window_partition_by_clause>] 
    [SERIES(...) [<window_partition_by_clause>]
    [<window_partition_by_clause>] [SERIES(...)] [<window_partition_by_clause>]
    [<window_partition_by_clause>] | [<window_partition_by_clause>]
    [<window_order_by_clause>] 
}) <expression> ::= <identifier>
```

**Syntax Elements**

- **expression**
  This parameter specifies the input data column.
- **ModeArgument**
  This parameter defines the applied extrapolation mode. The table below lists the possible values for `<ModeArgument>`.

```sql
<ModeArgument> ::= EXTRAPOLATION_NONE
               | EXTRAPOLATION_LINEAR
               | EXTRAPOLATION_CONSTANT
```

---

SAP HANA SQL Reference Guide for SAP HANA Platform

SQL Reference
<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTRAPOLATION_NONE</td>
<td>Prevents the extrapolation of leading and trailing nulls. Interpolation is performed on the values in the middle. Default value if &lt;ModeArgument&gt; is omitted. You do not need to specify the &lt;Value1Argument&gt; and &lt;Value2Argument&gt; parameters when using this mode. For example, an input of [null, null, 1, 2, null, null, 5, null, null] returns [null, null, 1, 2, 3, 4, 5, null, null].</td>
</tr>
<tr>
<td>EXTRAPOLATION_LINEAR</td>
<td>Performs linear extrapolation where &lt;Value1Argument&gt; is the minimum value and &lt;Value2Argument&gt; is the maximum value. During the extrapolation of leading and trailing nulls, this function checks that the extrapolation results are within range of the minimum and maximum values; otherwise, values are replaced by the minimum or maximum value, whichever is appropriate. An exception is thrown if &lt;Value1Argument&gt; is greater than &lt;Value2Argument&gt;.</td>
</tr>
<tr>
<td>EXTRAPOLATION_CONSTANT</td>
<td>Performs linear extrapolation where leading and trailing nulls values are replaced by the values specified by the &lt;Value1Argument&gt; and &lt;Value2Argument&gt; parameters, respectively. When &lt;Value1Argument&gt; is not specified or null, the first non-null value is used to replace leading null values. When &lt;Value2Argument&gt; is not specified or null, the last non-null is used to replace trailing null values.</td>
</tr>
</tbody>
</table>

Linear interpolation can be applied to all number types if they can be cast into a long double type, such as INT, LONG, DOUBLE, or FLOAT.

**Value1Argument and Value2Argument**

These parameters define arguments for the extrapolation and can be any numeric type.

```
<Value1Argument> ::= <identifier>
<Value2Argument> ::= <identifier>
```

**table_schema**

```
<table_schema> ::= [<table_schema>].<table_name>
<schema_name> ::= <unicode_name>
<table_name> ::= <identifier>
```

<table_schema> must be a base table name and not a correlation name.

**window_order_by_clause**

Determines the sort order. This expression must not contain any NULL values or duplicates. See the syntax of this clause located in the *Window Functions* topic.
Description

Operates on an entire series to produce a new series that replaces missing values by interpolating between adjacent non-NULL values and extrapolating any leading or trailing null values.

When using the SERIES TABLE syntax, the properties of the series table are used to determine the partitioning and ordering of rows before linear approximation is performed.

When using this function, the input is assumed to be equidistant. The SERIES definition must be equidistant.

When only null values are contained in the input, only null values appear in the output unless the null values are replaced by an extrapolation via the EXTRAPOLATION_CONSTANT mode.

When using the SERIES syntax, this function uses the properties in the following table to perform linear approximation.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NON-EQUIDISTANT</td>
<td>An error occurs if the series is non-equidistant.</td>
</tr>
<tr>
<td>MISSING ELEMENTS ALLOWED</td>
<td>When specified, there must be exactly one ORDER BY column that is compatible with the type of the series period column and the INCREMENT BY value.</td>
</tr>
<tr>
<td>PARTITION BY</td>
<td>If not specified, the SERIES KEY property of the SERIES syntax is used to construct the default PARTITION BY.</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>If not specified, the first PERIOD column is added as an ORDER BY.</td>
</tr>
</tbody>
</table>

Other SERIES properties, such as MINVALUE and MAXVALUE, are ignored.

Examples

Linear approximation without a series definition

If the SERIES definition were not specified in the example above, then the result would have been different. For the same SparseApproxTable, as populated above, the example below does not specify a series. The example below returns [-1, 0, 1, 2, 3, 4, 5, 6].

```sql
SELECT LINEAR_APPROX(val, 'EXTRAPOLATION_LINEAR') OVER (PARTITION BY ts_id ORDER BY DATE) AS approximated_value
FROM "SparseApproxTable";
```

In the example query above, the series is assumed to be dense and equidistant, so the date column is not taken into consideration when calculating the approximations.

Linear approximation of sparse series data

In the example below, the INCREMENT BY INTERVAL is set to 1 MONTH, meaning that at most one row is expected for each month. Apart from the null values, values are missing for several months in the table. The LINEAR_APPROX function always replaces the null values and does not insert new lines for missing values.
However, if the SERIES definition is specified in the SELECT statement, then LINEAR_APPROX considers the missing months when calculating the slope between two non-null values.

CREATE COLUMN TABLE "SparseApproxTable" (ts_id VARCHAR(20), date DAYDATE, val DOUBLE);
INSERT INTO "SparseApproxTable" VALUES('A','2013-11-01', null);
INSERT INTO "SparseApproxTable" VALUES('A','2014-01-01', null);
INSERT INTO "SparseApproxTable" VALUES('A','2014-02-05', 2);
INSERT INTO "SparseApproxTable" VALUES('A','2014-03-07', null);
INSERT INTO "SparseApproxTable" VALUES('A','2014-05-01', 5);
INSERT INTO "SparseApproxTable" VALUES('A','2014-07-27', 7);
INSERT INTO "SparseApproxTable" VALUES('A','2014-12-07', null);
INSERT INTO "SparseApproxTable" VALUES('A','2015-02-07', null);
SELECT LINEAR_APPROX(val, 'EXTRAPOLATION_LINEAR') OVER (SERIES (SERIES KEY(ts_id) EQUIDISTANT INCREMENT BY INTERVAL 1 MONTH MISSING ELEMENTS ALLOWED PERIOD FOR SERIES(date)) PARTITION BY ts_id) AS approximated_value FROM "SparseApproxTable";

<table>
<thead>
<tr>
<th>APPROXIMATED_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>9.6666666666666666</td>
</tr>
<tr>
<td>11</td>
</tr>
</tbody>
</table>

Linear approximation of series data

CREATE COLUMN TABLE "InterpolationTable" (TS_ID VARCHAR(20), date DAYDATE, val DOUBLE);
INSERT INTO "InterpolationTable" VALUES('A','2013-09-30', 1);
INSERT INTO "InterpolationTable" VALUES('A','2013-10-01', 2);
INSERT INTO "InterpolationTable" VALUES('A','2013-10-02', null);
INSERT INTO "InterpolationTable" VALUES('A','2013-10-03', 10);
SELECT date, val, LINEAR_APPROX (val, 'EXTRAPOLATION_LINEAR') OVER (PARTITION BY TS_ID ORDER BY date) AS LINEAR_APPROX FROM "InterpolationTable";

<table>
<thead>
<tr>
<th>DATE</th>
<th>VAL</th>
<th>MYRESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 30, 2013</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Oct 1, 2013</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Oct 2, 2013</td>
<td>?</td>
<td>6</td>
</tr>
</tbody>
</table>
Related Information

Expressions [page 56]
Window Functions and the Window Specification [page 424]

4.9.1.116 LN Function (Numeric)

Returns the natural logarithm of a number.

Syntax

LN(<number>)

Description

Returns the natural logarithm of the specified number.

Example

This example returns the natural logarithm of the number 9, which is 2.1972245773362196:

SELECT LN (9) "ln" FROM DUMMY;

Related Information

Numeric Data Types [page 48]
4.9.1.117 LOCALTOTUTC Function (Datetime)

A timestamp parameter holding the time to be converted between UTC and local time.

Syntax

```sql
LOCALTOTUTC (<time> [, <timezone> [, <timezone_dataset>]])
```

Syntax Elements

**time**

Specifies the time to be converted between UTC and local time.

```sql
<time> ::= <timestamp>
```

**timezone**

Specifies the timezone defining the local time. For a list of available timezones, see the TIMEZONES system view. If the local timezone is not explicitly specified, then the local timezone of the SAP HANA system is used.

```sql
<timezone> ::= <string_literal>
```

**timezone_dataset**

Specifies the dataset in which to search for the given timezone.

```sql
<timezone_dataset> ::= { sap | platform }
```

- `sap` searches in the dataset in the timezone definition tables. This is the default value. To use this value, import a timezone dataset (see SAP Note 1791342). If you use this value without importing a dataset, the function uses hardcoded fallback values. These fallback values are deprecated and will be removed in a future support package stack.
- `platform` searches in the dataset provided by the operating system.

Description

Converts the local time `<time>` from a timezone to the UTC(GMT) time.

The usage of local timestamps is discouraged; use UTC times instead. The use of local times or conversion between local time zones might require additional handling in the application code.
Examples

The following example returns the value 2012-01-01 06:00:00.0 for the UTC date and time:

```sql
SELECT LOCALTOUTC (TO_TIMESTAMP('2012-01-01 01:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'EST') "localtoutc" FROM DUMMY;
```

The following example returns the value 2012-01-01 06:00:00.0 for the UTC date and time:

```sql
SELECT LOCALTOUTC (TO_TIMESTAMP('2012-01-01 01:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'EST', 'sap') "localtoutc" FROM DUMMY;
```

Related Information

TIMEZONES System View [page 1692]
SAP Note 1791342

4.9.1.118 LOCATE Function (String)

Returns the position of a substring within a string.

Syntax

```
LOCATE( <haystack>, <needle>, [<start_position>], [ <occurrences> ] )
```

Description

The LOCATE function returns the position of a substring `<needle>` within a string `<haystack>`.

- If `<needle>` is not found within `<haystack>`, or if `<occurrences>` is set to less than 1, then 0 is returned.
- If `<haystack>`, `<needle>`, or `<occurrences>` is (explicitly) NULL, then NULL is returned.
- If `<occurrences>` is not specified, then the first matched position is returned. A setting of 1 for `<occurrences>` is the same as not specifying it, while a setting of `n` returns the `n`th match (or 0 if there are no more matches found).
- If `<start_position>` is a positive integer, or 0, or not specified, then the matching direction proceeds from left to right.
• If `<start_position>` is negative, then matching starts at the end of `<haystack>` and proceeds in the reverse direction (right to left). For example:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT LOCATE('AAA', 'A', -1) FROM &quot;DUMMY&quot;;</td>
<td>3 - the position of the last A</td>
</tr>
<tr>
<td>SELECT LOCATE('AAA', 'A', -2) FROM &quot;DUMMY&quot;;</td>
<td>2 - the position of the second to last A</td>
</tr>
<tr>
<td>SELECT LOCATE('AAA', 'A', -3) FROM &quot;DUMMY&quot;;</td>
<td>1 - the position of the third to last A</td>
</tr>
<tr>
<td>SELECT LOCATE('AAA', 'A', -4) FROM &quot;DUMMY&quot;;</td>
<td>0 - not found</td>
</tr>
<tr>
<td>SELECT LOCATE('ABABAC', 'A', -2) FROM &quot;DUMMY&quot;;</td>
<td>5 - the position of the first A starting from the second last position</td>
</tr>
</tbody>
</table>

• If a match is found, then the match position returned is always a positive number, even when `<start_position>` is a negative number.

**Examples**

The following example returns 1 because `<needle>` is an empty string.

```
SELECT LOCATE ('length in char', '') "locate" FROM DUMMY;
```

The following example returns the starting position (1) of `length` in the string `length in char`:

```
SELECT LOCATE ('length in char', 'length') "locate" FROM DUMMY;
```

The following example returns 0 because the search pattern `zchar` cannot be found in the given string:

```
SELECT LOCATE ('length in char', 'zchar') "locate" FROM DUMMY;
```

**Related Information**

*Character String Data Types [page 36]*
4.9.1.119  LOCATE_REXEXPR Function (String)

Searches a string for a regular expression pattern and returns an integer indicating the beginning position, or the ending position plus 1, of one occurrence of the matched substring.

Syntax

LOCATE_REXEXPR{  [  <regex_position_start_or_after>  ]  <pattern>  
  [  FLAG <flag>  ]  
  IN <regex_subject_string>  
  [  FROM <start_position>  ]  
  [  OCCURRENCE <regex_occurrence>  ]  
  [  GROUP <regex_capture_group>  ]  
}

Syntax Elements

regex_position_start_or_after
Searches a string for a regular expression pattern and returns an integer indicating the beginning position, or the ending position plus 1, of one occurrence of the matched substring.

<regex_position_start_or_after> ::= START | AFTER

If <regex_position_start_or_after> is not specified, then START is implicit.

pattern
A search pattern based on Perl Compatible Regular Expression (PCRE).

flag
The matching behavior of the function can be defined by the <flag> literal. The following options are available:

<flag> ::= 'i' | 'm' | 's' | 'x'

Flag options

<table>
<thead>
<tr>
<th>Flag option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Enables case-insensitive matching</td>
</tr>
<tr>
<td>m</td>
<td>Enables multiline mode, where the &lt;subject_string&gt; will be treated as multiple lines and the expression ^ and $ match just after or just before, respectively, a line terminator or the end of the input sequence</td>
</tr>
<tr>
<td>s</td>
<td>Enables the expression &lt;.&gt; as a wildcard to match any character, including a line terminator</td>
</tr>
<tr>
<td>x</td>
<td>Permits whitespace and comments in the pattern</td>
</tr>
</tbody>
</table>
SAP HANA uses the Perl-compatible Regular Expressions (PCRE) library to process regular expressions. Specifying 'U' (short for "ungreedy") inverts the "greediness" of quantifiers so that they are not greedy by default but become greedy only when followed by "?". Ungreedy matching can often perform faster because it finds the shorter match at times when it is only interesting to know whether there is any match.

For a full understanding of the "greedy" versus "ungreedy" matching behavior of the Perl-compatible Regular Expressions (PCRE) library, visit: https://www.pcre.org/original/doc/html/pcre-matching.html.

---

### Flag option | Description
---|---
**U** | SAP HANA uses the Perl-compatible Regular Expressions (PCRE) library to process regular expressions. Specifying 'U' (short for "ungreedy") inverts the "greediness" of quantifiers so that they are not greedy by default but become greedy only when followed by "?". Ungreedy matching can often perform faster because it finds the shorter match at times when it is only interesting to know whether there is any match.

---

**regex_subject_string**

<regex_subject_string> ::= <string>

Specifies a string in which to search for the regular expression pattern.

**start_position**

The <start_position> parameter is a positive integer and indicates the character of <regex_subject_string> where the search is started. If <start_position> is not a positive integer, then 0 is returned.

<start_position> ::= <positive_integer>

**regex_occurrence**

Specifies a positive integer to the occurrence of the <pattern> in <regex_subject_string>. The default is 1.

<regex_occurrence> ::= <integer>

If <regex_occurrence> is not a positive integer, then 0 is returned.

**regex_capture_group**

The <regex_capture_group> parameter is a non-negative integer and indicates the number of the captured substring's group by the regular expression. The default is 0.

<regex_capture_group> ::= <integer>

If <regex_capture_group> is a negative integer, then 0 is returned.

---

**Description**

Searches a string for a regular expression pattern and returns an integer indicating the beginning position, or the ending position plus 1, of one occurrence of the matched substring.

If any of the following parameters is NULL: <pattern>, <flag>, <regex_subject_string>, <start_position>, <regex_occurrence> or <regex_capture_group>, then the function returns NULL.
Example

This example returns the start position of the day part from the date value 20140401; it returns 7:

```
SELECT LOCATE_REGEXPR(START '([[:digit:]]{4})([[:digit:]]{2})([[:digit:]]{2})'
IN '20140401' GROUP 3) "locate_regexpr" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]

4.9.1.120 LOG Function (Numeric)

Returns the natural logarithm of a specified number and base.

Syntax

```
LOG(<base>, <number>)
```

Description

Returns the natural logarithm of a number specified by `<number>` and a base specified by `<base>`, where `<base>` must be a positive value other than 1, and `<number>` must be any positive value.

Example

The following example returns the natural logarithm for 2 base 10, which is 0.30102999566398114:

```
SELECT LOG (10, 2) "log" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]
4.9.1.121  LOWER Function (String)

Converts all characters in a string to lowercase.

**Syntax**

```sql
LOWER(<string>)
```

**Description**

Converts all characters in `<string>` to lowercase.

The LOWER function is identical to the LCASE function.

**Example**

This example converts the given string `AnT` to lowercase, and returns the value `ant`:

```sql
SELECT LOWER ('AnT') "lower" FROM DUMMY;
```

**Related Information**

LCASE Function (String) [page 239]
Character String Data Types [page 36]

4.9.1.122  LPAD Function (String)

Left-pads a string with spaces, or a specified pattern, to make a string of a specified number of characters in length.

**Syntax**

```sql
LPAD(<string>, <number> [, <pattern>])
```
Syntax Elements

- **string**
  Specifies a string to be padded.

- **number**
  Specifies the length to which to pad `<string>`. `<number>` must be an integer.

- **pattern**
  Specifies a string of characters to use for padding instead of spaces.

Description

Left-pads the end of `<string>` with spaces to make a string of `<number>` characters. If `<pattern>` is specified, then `<string>` is padded using sequences of the given characters until the required length is met.

If the length of `<string>` is greater than `<number>`, then no padding is performed and the resulting value is truncated from the right side to the length specified in `<number>`.

LPAD returns an empty string value if `<number>` is less than 1.

Example

The following example left-pads the start of string *end* with the pattern `12345` to make a string of 15 characters in length, and returns the value `123451234512end`:

```sql
SELECT LPAD ('end', 15, '12345') "lpad" FROM DUMMY;
```

In the following example, `<string>` is longer than `<n>`, so no padding is performed and the result is `<string>` truncated to the length of `<number>` (that is, `en`):

```sql
SELECT LPAD ('end', 2, '12345') "lpad" FROM DUMMY;
```

Related Information

- [Character String Data Types](page 36)
4.9.1.123  LTRIM Function (String)

Returns a string, trimmed of all leading spaces.

Syntax

LTRIM(<string> [, <remove_set>])

Description

Returns string <string>, trimmed of all leading spaces. If <remove_set> is specified, LTRIM removes all the characters contained in this set from the start of string <string>. This process continues until a character that is not in <remove_set> is reached.

<remove_set> is treated as a set of characters and not as a search string.

Example

This example removes all leading a and b characters from the given string and returns the value Aabend:

```
SELECT LTRIM ('babababAabend','ab') "ltrim" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]

4.9.1.124  MAP Function (Miscellaneous)

Searches for an expression within a set of values and returns a specified result.

Syntax

MAP(<expression>, <search_value>, <result> [, <search_value>, <result> [...] ] [, <default_result>])
Syntax Elements

**expression**

Specifies the expression to search for in any of the `<search>` values.

**search_value**

Specifies a value to search for `<expression>` in. You can specify more than one search value, and for each `<search_value>` value you can specify a corresponding `<result>`.

**result**

Specifies the result to return when `<expression>` is found in the corresponding `<search_value>` value.

**default_result**

Specifies the default result to return when `<expression>` is not found in any of the `<search_value>` values.

Description

Searches for `<expression>` within a set of search values and returns the corresponding result.

- If `<expression>` is not found and `<default_result>` is defined, then MAP returns `<default_result>`.
- If `<expression>` is not found and `<default_result>` is not defined, then MAP returns NULL.

`<search_value>` and corresponding `<result>` values must be specified as pairs.

Example

The following query searches for 2, finds it in the third `<search>/<result>` pair (2, 'Two'), and returns the configured `<result>` value Two.

```
SELECT MAP(2, 0, 'Zero', 1, 'One', 2, 'Two', 3, 'Three', 'Default') "map" FROM DUMMY;
```

The following searches for 99 in the `<search>` values, and, not finding it, returns the `<default_result>` value Default.

```
SELECT MAP(99, 0, 'Zero', 1, 'One', 2, 'Two', 3, 'Three', 'Default') "map" FROM DUMMY;
```

The following searches for 99 in the `<search>` values, and, not finding it, returns NULL because no `<default_result>` was specified.

```
SELECT MAP(99, 0, 'Zero', 1, 'One', 2, 'Two', 3, 'Three') "map" FROM DUMMY;
```
Related Information

Expressions [page 56]

4.9.1.125 MAX Function (Aggregate)

Returns the maximum value of the expression. This function can also be used as a window function.

Syntax

Aggregate function:

\[
\text{MAX( [ ALL | DISTINCT ] } \ <\text{expression}> \ )
\]

Window function:

\[
\text{MAX( } \ <\text{expression}> \ ) \ <\text{window_specification}>
\]

Syntax Elements

expression

Specifies the input data for the function.

window_specification

Defines a window on the data over which the function operates. For \(<\text{window_specification}>\), see Window Functions and the Window Specification [page 424].

Description

Result type based on input

<table>
<thead>
<tr>
<th>TINYINT</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>BIGINT</th>
<th>DECIMAL(p, s)</th>
<th>DECIMAL</th>
<th>REAL</th>
<th>DOUBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TINYINT</td>
<td>SMALLINT</td>
<td>INTEGER</td>
<td>BIGINT</td>
<td>DECIMAL(p, s)</td>
<td>DECIMAL</td>
<td>REAL</td>
<td>DOUBLE</td>
</tr>
</tbody>
</table>
**Example**

The following statements create a table with data for example purposes:

```sql
DROP TABLE "MyProducts";
CREATE COLUMN TABLE "MyProducts"(
    "Product_ID" VARCHAR(10),
    "Product_Name" VARCHAR(100),
    "Category" VARCHAR(100),
    "Quantity" INTEGER,
    "Price" DECIMAL(10,2),
    PRIMARY KEY ("Product_ID")
);
```

```sql
INSERT INTO "MyProducts" VALUES('P1','Shirts', 'Clothes', 32, 20.99);
INSERT INTO "MyProducts" VALUES('P2','Jackets', 'Clothes', 16, 99.49);
INSERT INTO "MyProducts" VALUES('P3','Trousers', 'Clothes', 30, 32.99);
INSERT INTO "MyProducts" VALUES('P4','Coats', 'Clothes', 5, 129.99);
INSERT INTO "MyProducts" VALUES('P5','Purse', 'Accessories', 3, 89.49);
```

The following example returns 129.99, the maximum price of the products in the MyProducts table:

```sql
SELECT MAX("Price") FROM "MyProducts";
```

**Related Information**

- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- Aggregate Functions [page 414]
- Expressions [page 56]

**4.9.1.126 MEDIAN Function (Aggregate)**

Finds the statistical median of an input expression with a numeric data type. This function can also be used as a window function.

**Syntax**

**Aggregate function:**

```sql
MEDIAN( <expression> )
```

**Window function:**

```sql
MEDIAN( <expression> ) <window_specification>
```
Syntax Elements

expression
Specifies the input data expression for the MEDIAN function.

window_specification
Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].

Description

The MEDIAN function finds the statistical median of an input expression with a numeric data type. Although it is grouped with the aggregate functions, its optional OVER clause positions it as a windows function as well.

Null values are eliminated. If there is an even number of elements, then the average of the two middle elements is returned. Otherwise, the middle element is returned.

The result type is the type that is selected for the expression "x/2" for an x value of the input data type.

Examples

**Median of integer input** The following example returns a median value of 2.

```sql
CREATE ROW TABLE T (class CHAR(10), date DAYDATE, val INT);
INSERT INTO T VALUES('A', '01.01.1972', 1);
INSERT INTO T VALUES('A', '02.01.1972', 3);
INSERT INTO T VALUES('A', '03.01.1972', null);
INSERT INTO T VALUES('A', '04.01.1972', 2);
SELECT MEDIAN(val) "median value" FROM T;
```

If the number of non-null values is even, then the average of the two middle values is returned. For the following example, the average of 2 and 3 is returned. Since the input and output types are the same, the integer is rounded. The returned result is 3.

```sql
INSERT INTO T VALUES('A', '05.01.1972', 4);
SELECT MEDIAN(val) "median value" FROM T;
```

**Median of double input** The following example uses double values instead of integers. The returned result is 2.5.

```sql
CREATE ROW TABLE T (TS_ID CHAR(10), date DAYDATE, val DOUBLE);
INSERT INTO T VALUES('A', '01.01.1972', 1.0);
INSERT INTO T VALUES('A', '02.01.1972', 3.0);
INSERT INTO T VALUES('A', '03.01.1972', null);
INSERT INTO T VALUES('A', '04.01.1972', 2.0);
INSERT INTO T VALUES('A', '05.01.1972', 4.0);
SELECT MEDIAN(val) "median value" FROM T;
```
**Median as a window function**  The following example uses double values instead of integers.

```sql
CREATE ROW TABLE T (TS_ID CHAR(10), date DAYDATE, val DOUBLE);
INSERT INTO T VALUES('A', '01.01.1972', 1.0);
INSERT INTO T VALUES('A', '02.01.1972', 3.0);
INSERT INTO T VALUES('A', '03.01.1972', null);
INSERT INTO T VALUES('A', '04.01.1972', 2.0);
INSERT INTO T VALUES('A', '04.01.1972', 4.0);
SELECT MEDIAN(val) OVER (PARTITION BY TS_ID ) AS WF1 FROM T;
```

The returned result is:

```
WF1
2.5
2.5
2.5
2.5
2.5
```

**Median of sliding window (GROUPS BETWEEN)**  Both of the SELECT statements in the following example produce identical results.

```sql
CREATE ROW TABLE T (TS_ID CHAR(10), date DAYDATE, val DOUBLE);
INSERT INTO T VALUES('A', '01.01.1972', 1.0);
INSERT INTO T VALUES('A', '02.01.1972', 3.0);
INSERT INTO T VALUES('A', '03.01.1972', null);
INSERT INTO T VALUES('A', '04.01.1972', 2.0);
INSERT INTO T VALUES('A', '04.01.1972', 4.0);
SELECT MEDIAN(val) OVER (PARTITION BY TS_ID ORDER BY date) AS WF2A FROM T;
SELECT MEDIAN(val) OVER (PARTITION BY TS_ID ORDER BY date GROUPS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS WF2B FROM T;
```

The returned result is:

```
WF2A
1
1
2
2
2
2
2.5
```
Median of sliding window (ROWS BETWEEN) The following example uses ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

```sql
CREATE ROW TABLE T (TS_ID CHAR(10), date DAYDATE, val DOUBLE);
INSERT INTO T VALUES('A', '01.01.1972', 1.0);
INSERT INTO T VALUES('A', '02.01.1972', 3.0);
INSERT INTO T VALUES('A', '03.01.1972', null);
INSERT INTO T VALUES('A', '04.01.1972', 2.0);
INSERT INTO T VALUES('A', '04.01.1972', 4.0);
SELECT MEDIAN(val) OVER (PARTITION BY TS_ID ORDER BY date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS WF3 FROM T;
```

The returned result is:

```
WF3
1
1
1
1
1
1
2
2
2
2
2
2
2
2
2.5
2
2.5
2
2.5
2
```
4.9.127 MEMBER_AT Function (Array)

Returns values from a specified array position.

Syntax

```
MEMBER_AT(<array_value_expression>, <position> [, <default_value>])
```

Description

Accesses an array element at the specified ordinal position and returns the value. If the <array_value_expression> does not have an element at the specified position, then it returns the <default_value> specified or the value of the specified <position> if <default_value> is not specified. If <default_value> is not specified, and <position> is greater than the cardinality of the <array_value_expression>, then NULL is returned.

Example

The following example creates a table and inserts arrays into it:

```
CREATE COLUMN TABLE ARRAY_TEST (IDX INT, VAL INT ARRAY);
INSERT INTO ARRAY_TEST VALUES (1, ARRAY(1, 2, 3));
INSERT INTO ARRAY_TEST VALUES (2, ARRAY(10, 20, 30, 40));
```

The following example returns the value in position 4 of each array in the table. The first array does not have a position 4, so NULL is returned:

```
SELECT MEMBER_AT(VAL, 4) "member_at" FROM ARRAY_TEST;
```

```
member_at
? NULL
```
The following example returns the value in position 4 of each array in the table. The first array does not have a position 4, so 1 is returned <position> is greater than the cardinality of the array:

```
SELECT MEMBER_AT(VAL, 4, 1) "member_at" FROM ARRAY_TEST;
```

<table>
<thead>
<tr>
<th>member_at</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>40</td>
</tr>
</tbody>
</table>

### Related Information

**Expressions [page 56]**

### 4.9.1.128 MIMETYPE Function (Fulltext)

Returns the MIME type of a value in a column that has a fulltext index.

#### Syntax

```
MIMETYPE(<column_name>)
```

#### Syntax Elements

- **column_name** Specifies the column values to test for the MIME type.

```
<column_name> ::= <identifier>
```

#### Description

Returns the MIME type of the corresponding column value. This function requires a fulltext index on the input column.
Example

The following example creates a table with two differing types of text content, plain text and HTML:

```sql
CREATE COLUMN TABLE T (CONTENT TEXT FAST PREPROCESS OFF);
INSERT INTO T VALUES('This is an example');
INSERT INTO T VALUES('<html>This is an example</html>');
```

Select the content and detect the MIME type of the entries:

```sql
SELECT MIMETYPE(CONTENT), CONTENT FROM T;
```

<table>
<thead>
<tr>
<th>MIMETYPE(CONTENT)</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>text/plain</td>
<td>This is an example</td>
</tr>
<tr>
<td>text/html</td>
<td>&lt;html&gt;This is an example&lt;/html&gt;</td>
</tr>
</tbody>
</table>

4.9.1.129 MIN Function (Aggregate)

Returns the minimum value of the expression. This function can also be used as a window function.

Syntax

Aggregate function:

```sql
MIN([ ALL | DISTINCT ] <expression> )
```

Window function:

```sql
MIN(<expression>) <window_specification>
```

Syntax Elements

expression

Specifies the input data for the function.

window_specification

Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].
Description

Result type based on input

<table>
<thead>
<tr>
<th>Type</th>
<th>TINYINT</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>BIGINT</th>
<th>DECIMAL (p, s)</th>
<th>DECIMAL</th>
<th>REAL</th>
<th>DOUBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TINYINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL(p)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL(s)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

The following statements create a table with data for example purposes:

```sql
DROP TABLE "MyProducts";
CREATE TABLE "MyProducts"(
  "Product_ID" VARCHAR(10),
  "Product_Name" VARCHAR(100),
  "Category" VARCHAR(100),
  "Quantity" INTEGER,
  "Price" DECIMAL(10,2),
  PRIMARY KEY ("Product_ID")
);

INSERT INTO "MyProducts" VALUES('P1','Shirts', 'Clothes', 32, 20.99);
INSERT INTO "MyProducts" VALUES('P2','Jackets', 'Clothes', 16, 99.49);
INSERT INTO "MyProducts" VALUES('P3','Trousers', 'Clothes', 30, 32.99);
INSERT INTO "MyProducts" VALUES('P4','Coats', 'Clothes', 5, 129.99);
INSERT INTO "MyProducts" VALUES('P5','Purse', 'Accessories', 3, 89.49);
```

The following example returns 20.99, the minimum price of the products in the MyProducts table:

```sql
SELECT MIN("Price") FROM "MyProducts";
```

Related Information

Window Functions and the Window Specification [page 424]
Window Aggregate Functions [page 428]
Aggregate Functions [page 414]
Expressions [page 56]
4.9.1.130  MINUTE Function (Datetime)

Returns an integer representation of the minute for the specified time.

Syntax

```
MINUTE(<time>)
```

Description

Returns an integer representation of the minute for time `<time>`.

Example

The following example returns the value 34 as the minute for the specified time:

```
SELECT MINUTE ('12:34:56') "minute" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]

4.9.1.131  MOD Function (Numeric)

Returns the remainder of a specified number divided by a specified divisor.

Syntax

```
MOD(<number>, <divisor>)
```
Description

Returns the remainder of a number <number> divided by a divisor <divisor>.

When <number> is negative, this function acts differently to the standard computational modulo operation. The following list shows examples of what the MOD function returns as the result:

- If <divisor> is zero, then <number> is returned.
- If <number> is greater than 0 and <number> is less than <divisor>, then <number> is returned.
- If <number> is less than 0 and <number> is greater than <divisor>, then <number> is returned.
- In cases other than those mentioned above, the remainder of the absolute value of <number> divided by the absolute value of <divisor> is used to calculate the remainder. If <number> is less than 0, then the returned remainder from MOD is a negative number, and if <number> is greater than 0, then the returned remainder from MOD is a positive number.

Example

The following example returns the value 3:

```sql
SELECT MOD (15, 4) "modulus" FROM DUMMY;
```

The following example returns the value -3:

```sql
SELECT MOD (-15, 4) "modulus" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.132  MONTH Function (Datetime)

Returns the number of the month from the specified date.

Syntax

```sql
MONTH(<date>)
```
Description

Returns the number of the month from date <date>.

Example

The following example returns the value 5, the month for the date specified:

```
SELECT MONTH ('2011-05-30') "month" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]

4.9.1.133  MONTHNAME Function (Datetime)

Returns the name of the month for the specified date.

Syntax

```
MONTHNAME(<date>)
```

Description

Returns the name of the month for date <date>.

Example

The following example returns the value MAY, the month name of the specified date:

```
SELECT MONTHNAME ('2011-05-30') "monthname" FROM DUMMY;
```
4.9.1.134 MONTHS_BETWEEN Function (Datetime)

Computes the number of months between two dates.

Syntax

MONTHS_BETWEEN(<date_1>, <date_2>)

Description

Computes the number of months between <date_1> and <date_2>. Returns NULL if either of <date_1> or <date_2> is NULL.

Examples

The following example returns 2 for the months between the two dates:

```
SELECT MONTHS_BETWEEN(TO_DATE('2003-01-01'), TO_DATE('2003-03-14'))
"months_between" FROM DUMMY;
```

The following example returns -8 for the months between the two dates:

```
SELECT MONTHS_BETWEEN(TO_DATE('2003-10-03'), TO_DATE('2003-01-14'))
"months_between" FROM DUMMY;
```

The following example returns -9 for the months between the two dates:

```
SELECT MONTHS_BETWEEN(TO_DATE('2003-10-15'), TO_DATE('2003-01-14'))
"months_between" FROM DUMMY;
```

The following example returns -8 for the months between the two dates:

```
SELECT MONTHS_BETWEEN(TO_DATE('2003-10-13'), TO_DATE('2003-01-14'))
"months_between" FROM DUMMY;
```
### 4.9.1.135 NANO100_BETWEEN Function (Datetime)

Computes the time difference between two dates to the precision of 0.1 microseconds.

**Syntax**

```
NANO100_BETWEEN(<timestamp_1>, <timestamp_2>)
```

**Description**

Computes the time difference between date arguments `<timestamp_1>` and `<timestamp_2>`, to the precision of 0.1 microseconds.

The precision for arguments of data type seconddate are truncated down to seconds. Use arguments of data type timestamp instead.

**Example**

The following example returns 477 as the time difference between the two specified timestamps:

```
SELECT NANO100_BETWEEN ('2021-01-31 00:00:00.0000176', '2021-01-31 00:00:00.0000653') "nano100 between" FROM DUMMY;
```

**Related Information**

Datetime Data Types [page 40]
4.9.1.136 NCHAR Function (String)

Returns the Unicode character with the specified code number.

Syntax

NCHAR(<number>)

Description

Returns the Unicode character with the specified code number <number>.

Example

This example returns the Unicode character (a) with the code number 65:

SELECT NCHAR (65) "nchar" FROM DUMMY;

Related Information

Character String Data Types [page 36]

4.9.1.137 NDIV0 Function (Numeric)

Returns 0 when divided by 0; otherwise, it returns the result of the division.

Syntax

NDIV0( <numerator>, <denominator> )
**Description**

Returns 0 if `<denominator>` is 0. Otherwise, it returns the division of the numerator by the denominator. `<numerator>` and `<denominator>` can be any numeric value. The function allows you to avoid divided-by-zero error messages and can continue a calculation with a defined result.

**Example**

The following example uses the NDIV0 function to allow the revenue calculation to complete without a divide by zero error:

```
SELECT Revenue, Quantity, NDIV0(Revenue, Quantity) as AverageRevenue FROM DUMMY;
```

<table>
<thead>
<tr>
<th>Revenue</th>
<th>Quantity</th>
<th>AverageRevenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>4</td>
<td>250</td>
</tr>
<tr>
<td>2000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2500</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3000</td>
<td>5</td>
<td>600</td>
</tr>
</tbody>
</table>

### 4.9.1.138 NEXT_DAY Function (Datatime)

Returns the date of the day following the specified date.

**Syntax**

```
NEXT_DAY(<date>)
```

**Description**

Returns the date of the day following the specified date.
Example

The following example returns 2010-01-01 as the day after 2009-12-31:

```
SELECT NEXT_DAY (TO_DATE ('2009-12-31', 'YYYY-MM-DD')) "next day" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]

4.9.1.139 NEWUID Function (Miscellaneous)

Creates a unique identifier within an SAP HANA database.

Syntax

```
NEWUID()
```

description

The NEWUID function acts as an alternative to the SYSUUID function. The SYSUUID function generates a universally unique identifier, whereas the NEWUID function generates a unique identifier within one SAP HANA database. Both functions return VARBINARY(16).

Example

The following fictitious example returns a unique value, such as C15F6DB61C6A2B00F0000001EC2E2F00:

```
SELECT NEWUID() FROM DUMMY;
```
4.9.1.140 NORMALIZE Function (String)

Applies a normalizing format to a character value expression and returns a normalized string.

Syntax

```sql
NORMALIZE( <character_value_expression> [ , <normalized_format> ] )
```

Syntax Elements

- `character_value_expression`
  Specifies the VARCHAR or NVARCHAR string to normalize.

- `normalized_format`
  Specifies the encoding format to use for the result. The default format is NFC.

```
<normalized_format> ::=  NFC
                        | NFD
                        | NFKC  | NFKD
```

The four Unicode normalization forms are as follows:

<table>
<thead>
<tr>
<th>Form</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalization Form D (NFD)</td>
<td>Specifies canonical decomposition.</td>
</tr>
<tr>
<td>Normalization Form C (NFC)</td>
<td>Specifies canonical decomposition followed by canonical composition.</td>
</tr>
<tr>
<td>Normalization Form KD (NFKD)</td>
<td>Specifies compatibility decomposition.</td>
</tr>
<tr>
<td>Normalization Form KC (NFKC)</td>
<td>Specifies compatibility decomposition followed by canonical composition.</td>
</tr>
</tbody>
</table>

Description

The Unicode standard defines two formal types, which are canonically and compatibility equivalent to the same characters by four normalization forms. The Unicode normalization algorithm puts all combined marks in a specified order. Decomposition only or decomposition followed by the composition rule is applied to transform each string. The details can be found from the following Unicode Standard Annex #15: http://www.unicode.org/reports/tr15/
Example

The examples in this section use the NORMALIZE function to return the normalized equivalent of `<character_value_expression>` based on the `<normalized_format>` form.

The following is an example of canonical equivalence:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SELECT NORMALIZE('Å') FROM DUMMY;</code></td>
<td>Å (0x00C5)</td>
</tr>
<tr>
<td><code>SELECT NORMALIZE('A °') FROM DUMMY;</code></td>
<td>A</td>
</tr>
<tr>
<td><code>SELECT NORMALIZE('Å', 'NFC') FROM DUMMY;</code></td>
<td>Å (0x00C5)</td>
</tr>
<tr>
<td><code>SELECT NORMALIZE('Å', 'NFD') FROM DUMMY;</code></td>
<td>A °(0x0041 0x030A)</td>
</tr>
<tr>
<td><code>SELECT NORMALIZE('A °', 'NFC') FROM DUMMY;</code></td>
<td>Å (0x00C5)</td>
</tr>
<tr>
<td><code>SELECT NORMALIZE('A °', 'NFD') FROM DUMMY;</code></td>
<td>A °(0x0041 0x030A)</td>
</tr>
</tbody>
</table>

**Note**

The example is illustrative and does not work as shown with copy and paste.

The following is an example of compatibility equivalence:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SELECT NORMALIZE('ﻉ', 'NFKC') FROM DUMMY;</code></td>
<td>ﻊ (positional variants)</td>
</tr>
<tr>
<td><code>SELECT NORMALIZE('ﻊ', 'NFKC') FROM DUMMY;</code></td>
<td>ﻊ (positional variants)</td>
</tr>
<tr>
<td><code>SELECT NORMALIZE('①', 'NFKC') FROM DUMMY;</code></td>
<td>1 (circled variants)</td>
</tr>
<tr>
<td><code>SELECT NORMALIZE('ｶ', 'NFKC') FROM DUMMY;</code></td>
<td>ﾚ (width variants)</td>
</tr>
</tbody>
</table>
### Related Information

Expressions [page 56]

### 4.9.1.141  NOW Function (Datetime)

Returns the current timestamp.

#### Syntax

```sql
NOW()
```

#### Description

Returns the current timestamp.

#### Example

The following example returns a value similar to `2010-01-01 16:34:19.894` for the current timestamp:

```sql
SELECT NOW () "now" FROM DUMMY;
```

### Related Information

Datetime Data Types [page 40]
4.9.1.142  NTH_VALUE Function (Aggregate)

Returns the value of an element at a specific position in an expression. This function can also be used as a window function.

Syntax

Aggregate function:

\[
\text{NTH\_VALUE( } <\text{expression}>, <\text{position}> <\text{order\_by\_clause}> \text{ )}
\]

Window function:

\[
\text{NTH\_VALUE( } <\text{expression}>, <\text{position}> <\text{order\_by\_clause}> \text{ ) } <\text{window\_specification}>
\]

Syntax Elements

\text{expression}

Specifies the expression where the position of the element occurs.

\text{position}

Specifies the position of the element in \text{expression}.

\text{order\_by\_clause}

Specifies the sort order of the input rows.

\[
<\text{order\_by\_clause}> ::= \text{ORDER BY } <\text{order\_by\_expression}> [, <\text{order\_by\_expression}> [...]]
\]

\[
<\text{order\_by\_expression}> ::= <\text{column\_name}> [ \text{ASC | DESC} ] [ \text{NULLS FIRST | NULLS LAST} ] [ <\text{collate\_clause}> ]
\]

\[
| <\text{column\_position}> [ \text{ASC | DESC} ] [ \text{NULLS FIRST | NULLS LAST} ] [ <\text{collate\_clause}> ]
\]

\[
<\text{collate\_clause}> ::= \text{COLLATE } <\text{collation\_name}>
\]

\text{<collate\_clause> specifies the collation to use for ordering values in the results. <collate\_clause> can only be used on columns defined as NVARCHAR or VARCHAR. <collation\_name> is one of the supported collation names listed in the COLLATIONS system view.}

\text{window\_specification}

Defines a window on the data over which the function operates. For \text{<window\_specification>}, see Window Functions and the Window Specification [page 424].
Description

Returns the value of the element at position specified by `<position>` in `<expression>` as ordered by `<order_by_clause>`. Null is returned if the value is NULL or if the value of `<position>` is greater than the number of elements in `<expression>`. An error is raised if `<position>` is less than or equal to 0. The output of NTH_VALUE function can be non-deterministic among tied values.

Example

The example below returns the second value in column `COL1`: 700, when the table is ordered by `COL2`.

```sql
CREATE ROW TABLE T (COL1 DOUBLE, COL2 DOUBLE);
INSERT INTO T VALUES(900, 10);
INSERT INTO T VALUES(400, 50);
INSERT INTO T VALUES(700, 30);
INSERT INTO T VALUES(200, 40);
SELECT NTH_VALUE (COL1, 2 ORDER BY COL2) FROM T;
```

Example for Spatial Data Type (ST_Geometry, ST_Point)

The example below returns the nth value, `POINT (3 3)`, in binary format, when the table is ordered by ID:

```sql
CREATE TABLE TAB (ID INT, SHAPE ST_Geometry(4326));
INSERT INTO TAB VALUES(1, ST_GeomFromText('POINT(1 1)', 4326));
INSERT INTO TAB VALUES(2, ST_GeomFromText('POINT(2 2)', 4326));
INSERT INTO TAB VALUES(3, ST_GeomFromText('POINT(3 3)', 4326));
INSERT INTO TAB VALUES(4, ST_GeomFromText('POINT(4 4)', 4326));
SELECT NTH_VALUE(SHAPE, 3 ORDER BY ID) FROM TAB;
```

To return a result as data type NVARCHAR, use the following statement:

```sql
SELECT NTH_VALUE(CAST(SHAPE AS NVARCHAR), 3 ORDER BY ID) FROM TAB;
```

Related Information

COLLATIONS System View [page 1520]
Window Functions and the Window Specification [page 424]
Window Aggregate Functions [page 428]
SAP HANA Spatial Reference
4.9.1.143 NTILE Function (Window)

Distributes rows into a specified number of buckets and assigns the bucket number starting from 1 to each row in the bucket.

Syntax

\[
\text{NTILE}( \text{<number_of_buckets}> ) \text{ <window_specification>}
\]

Syntax Elements

number_of_buckets

Specifies the number of buckets to distribute rows to.

window_specification

Defines a window on the data over which the function operates. For <window_specification>, see Window Functions and the Window Specification [page 424].

Description

The output of NTILE function can be non-deterministic among tie values.

Examples

```
CREATE ROW TABLE T (class CHAR(10), val INT, offset INT);
INSERT INTO T VALUES('A', 1, 1);
INSERT INTO T VALUES('A', 3, 3);
INSERT INTO T VALUES('A', 5, null);
INSERT INTO T VALUES('A', 5, 2);
INSERT INTO T VALUES('A', 10, 0);
INSERT INTO T VALUES('B', 1, 3);
INSERT INTO T VALUES('B', 1, 1);
INSERT INTO T VALUES('B', 7, 1);
SELECT class, val,
    NTILE(3) OVER (PARTITION BY class ORDER BY val) AS NTILE,
    FIRST_VALUE(val) OVER (PARTITION BY class ORDER BY val) AS first,
    LAST_VALUE(val) OVER (PARTITION BY class ORDER BY val) AS last,
    NTH_VALUE(val, 4) OVER (PARTITION BY class ORDER BY val) AS nth
FROM T;
```
### Related Information

Expressions [page 56]
Window Functions and the Window Specification [page 424]

#### 4.9.1.144 NULLIF Function (Miscellaneous)

Determines whether two expressions are equal.

**Syntax**

```
NULLIF(<expression1>, <expression2>)
```

**Description**

NULLIF compares the values of two expressions and returns NULL if `<expression1>` equals `<expression2>`.

If `<expression1>` does not equal `<expression2>`, then NULLIF returns `<expression1>`.

If `<expression2>` is NULL, then NULLIF returns `<expression1>`.

NULLIF returns the same data type as `<expression1>`. However, if `<expression1>` is either TINYINT or SMALLINT, then NULLIF returns an INTEGER.
Example

The following query returns `diff`.

```sql
SELECT NULLIF ('diff', 'same') "nullif" FROM DUMMY;
```

The following query returns `NULL`.

```sql
SELECT NULLIF('same', 'same') "nullif" FROM DUMMY;
```

4.9.1.145 NUMTOHEX Function (String)

Converts a numeric value to a hexadecimal value.

Syntax

```sql
NUMTOHEX(<integer> [, <integer>])
```

Syntax Elements

- `integer`
  Specifies the numeric value to be converted to a hexadecimal value.

  The second parameter controls the length of the target type. If only one parameter is provided, then the length depends on the input type. The second parameter is limited to a 16 hex string length of BIGINT type.

Description

Converts a numeric value to a hexadecimal value as a VARCHAR data type. The input value can be a TINYINT, SMALLINT, BIGINT, or INTEGER data type.

Example

The following example converts the numeric value `c1` to a hexadecimal VARCHAR value, `3039`:

```sql
CREATE TABLE t1 (c1 BIGINT);
INSERT INTO t1 VALUES(9223372036854775807);
```
4.9.1.146 OCCURRENCES_REGEXPR Function (String)

Returns the number of matches of a regular expression search within a string.

Syntax

```sql
OCCURRENCES_REGEXPR( <pattern> [ FLAG <flag> ] IN <regex_subject_string> [ FROM <start_position> ] )
```

Syntax Elements

**pattern**

Specifies a search pattern based on Perl Compatible Regular Expression (PCRE).

```sql
<pattern> ::= !!Perl Compatible Regular Expression
```

**flag**

Specifies the matching behavior of the function and can be defined by the flag literal.

```sql
<flag> ::= { 'i' | 'm' | 's' | 'x' }
```

Flag options

<table>
<thead>
<tr>
<th>Flag option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Enables case-insensitive matching</td>
</tr>
<tr>
<td>m</td>
<td>Enables multiline mode, where the <code>&lt;subject_string&gt;</code> will be treated as multiple lines and the expression <code>^</code> and <code>$</code> match just after or just before, respectively, a line terminator or the end of the input sequence</td>
</tr>
<tr>
<td>s</td>
<td>Enables the expression <code>&lt;&gt;</code> as a wildcard to match any character, including a line terminator</td>
</tr>
<tr>
<td>x</td>
<td>Permits whitespace and comments in the pattern</td>
</tr>
</tbody>
</table>
SAP HANA uses the Perl-compatible Regular Expressions (PCRE) library to process regular expressions. Specifying 'U' (short for "ungreedy") inverts the "greediness" of quantifiers so that they are not greedy by default but become greedy only when followed by "?". Ungreedy matching can often perform faster because it finds the shorter match at times when it is only interesting to know whether there is any match.

For a full understanding of the "greedy" versus "ungreedy" matching behavior of the Perl-compatible Regular Expressions (PCRE) library, visit: https://www.pcre.org/original/doc/html/pcre-matching.html.

### regex_subject_string

Specifies the string to search within. If `<regex_subject_string>` is empty, then the result is empty.

```
<regex_subject_string> ::= <string>
```

### start_position

Specifies a positive integer that indicates the character of `<regex_subject_string>` where the search is started. If `<start_position>` is not a positive integer, then -1 is returned.

```
<start_position> ::= <positive_integer>
```

### Description

Returns the number of matches of a regular expression search within a string.

### Example

The following example returns the number of occurrences of digits in the specified string, 'a1b2', and returns the value 2:

```
SELECT OCCURRENCES_REGEXPR('([[:digit:]])' IN 'a1b2') "occurrences_regexpr" FROM DUMMY;
```
4.9.1.147 PERCENT_RANK Function (Window)

Calculates the percentage of values that are either less than or greater than the current value, per the ORDER BY specification for the window.

Syntax

PERCENT_RANK() <window_specification>

Syntax Elements

<window_specification>
Defines a window on the data over which the function operates. For <window_specification>, see Window Functions and the Window Specification [page 424].

Description

For any row in group of rows in a result set, PERCENT_RANK calculates the percentage of values that are either less than or greater than the current value, per the ORDER BY specification for the window.

PERCENT_RANK returns (fractional) integers starting at 0 and ending with 1, as defined by the sort order (ASC or DESC) in the ORDER BY clause.

The relative rank of a row R is defined as (RANK()-1)/(NR-1), where NR is defined to be the number of rows in the window partition of R.

Examples

In the this example, the rows are partitioned by ProdName, creating a group of rows for each product name. For each row, the PERCENT_RANK column expresses the rank of the sales value for that row as a percentage. Specifically, because ORDER BY (Sales ASC) is specified, PERCENT_RANK returns the percentage of values in the same group that have a lower value.

CREATE ROW TABLE ProductSales(ProdName VARCHAR(50), Type VARCHAR(20), Sales INT);
INSERT INTO ProductSales VALUES('Tee Shirt','Plain',21);
INSERT INTO ProductSales VALUES('Tee Shirt','Lettered',22);
INSERT INTO ProductSales VALUES('Tee Shirt','Team logo',30);
INSERT INTO ProductSales VALUES('Hoodie','Plain',60);
INSERT INTO ProductSales VALUES('Hoodie','Lettered',65);
INSERT INTO ProductSales VALUES('Hoodie','Team logo',80);
INSERT INTO ProductSales VALUES('Ballcap','Plain',8);

INSERT INTO ProductSales VALUES('Ballcap','Lettered',40);
INSERT INTO ProductSales VALUES('Ballcap','Team logo',27);
SELECT ProdName, Type, Sales,
PERCENT_RANK() OVER (PARTITION BY ProdName ORDER BY Sales ASC) AS Percent_Rank
FROM ProductSales
ORDER BY Sales DESC;

<table>
<thead>
<tr>
<th>PRODNAME</th>
<th>Type</th>
<th>SALES</th>
<th>PERCENT_RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hoodie</td>
<td>Team logo</td>
<td>80</td>
<td>1</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Vintage</td>
<td>67</td>
<td>0.6666666666666666</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Lettered</td>
<td>65</td>
<td>0.3333333333333333</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Plain</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>Ballcap</td>
<td>Lettered</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Team logo</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>Ballcap</td>
<td>Team logo</td>
<td>27</td>
<td>0.5</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Lettered</td>
<td>22</td>
<td>0.5</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Plain</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>Ballcap</td>
<td>Plain</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

**Related Information**

Window Functions and the Window Specification [page 424]
Window Aggregate Functions [page 428]
Aggregate Functions [page 414]
RANK Function (Window) [page 298]

**4.9.1.148 PERCENTILE_CONT Function (Window)**

Returns an interpolated value using the percentile value of a constant.

**Syntax**

PERCENTILE_CONT( <constant_literal> ) WITHIN GROUP ( ORDER BY <expression> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] ) <window_specification>
**Syntax Elements**

`window_specification`

 Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].

**Description**

Returns an interpolated value using the percentile value of `<constant_literal>`.

A continuous distribution of the values of `<expression>` is assumed. Only a single `<expression>` can be specified in ORDER BY clause; this expression must evaluate to a numeric type. The `<constant_literal>` should be a numeric value between, and including, 0 to 1, because it is a percentile value. In the interpolation, nulls are discarded in the calculation, and all indexes are 1 based. The details of the interpolation are as follows:

\[
\begin{align*}
NR & = \text{the number of rows in the window partition} \\
P & = \text{the specified percentile value}\* \\
INDEX1 & = \text{floor}(1 + (P \times (NR - 1))) \\
INDEX2 & = \text{ceil}(1 + (P \times (NR - 1))) \\
COEF1 & = INDEX2 - (1 + (P \times (NR - 1))) \\
COEF2 & = INDEX1 - (1 + (P \times (NR - 1))) \\
\end{align*}
\]

if `INDEX1 == INDEX2`
then returns (value at `INDEX1`)
else returns (\( (\text{value at } INDEX1) \times COEF1 + (\text{value at } INDEX2 ) \times COEF2 \))

**Examples**

```sql
DROP TABLE p_cont;
CREATE ROW TABLE p_cont (class CHAR(10), val INT, offset INT);
INSERT INTO p_cont VALUES('A', 1, 1);
INSERT INTO p_cont VALUES('A', 3, 3);
INSERT INTO p_cont VALUES('A', 5, null);
INSERT INTO p_cont VALUES('A', 5, 2);
INSERT INTO p_cont VALUES('A', 10, 0);
INSERT INTO p_cont VALUES('B', 1, 3);
INSERT INTO p_cont VALUES('B', 1, 1);
INSERT INTO p_cont VALUES('B', 7, 1);

SELECT class, val,
PERCENTILE_CONT(0.125) WITHIN GROUP (ORDER BY val) OVER (PARTITION BY class) AS pc1,
PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY val) OVER (PARTITION BY class) AS pc2,
PERCENTILE_CONT(0.875) WITHIN GROUP (ORDER BY val) OVER (PARTITION BY class) AS pc3
FROM p_cont;
```
### Related Information

Expressions [page 56]
Window Functions and the Window Specification [page 424]
PERCENTILE_DISC Function (Window) [page 289]

### 4.9.1.149 PERCENTILE_DISC Function (Window)

Returns the first value whose cumulative distribution value is greater than or equal to the percentile value of a constant.

### Syntax

```
PERCENTILE_DISC(<constant_literal>) WITHIN GROUP (ORDER BY <expression>[ASC | DESC][NULLS FIRST | NULLS LAST]) <window_specification>
```

### Syntax Elements

**window_specification**

Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].

---

<table>
<thead>
<tr>
<th>CLASS</th>
<th>VAL</th>
<th>PC1</th>
<th>PC2</th>
<th>PC3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2.0</td>
<td>5.0</td>
<td>7.5</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2.0</td>
<td>5.0</td>
<td>7.5</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>2.0</td>
<td>5.0</td>
<td>7.5</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>2.0</td>
<td>5.0</td>
<td>7.5</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>2.0</td>
<td>5.0</td>
<td>7.5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1.0</td>
<td>1.0</td>
<td>5.5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1.0</td>
<td>1.0</td>
<td>5.5</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>1.0</td>
<td>1.0</td>
<td>5.5</td>
</tr>
</tbody>
</table>
Description

Returns the first value whose cumulative distribution value is greater than or equal to the percentile value of `<constant_literal>`.

A discrete distribution of the values of `<expression>` is assumed. Only a single `<expression>` can be specified in ORDER BY clause; this expression must evaluate to a numeric type. The `<constant_literal>` should be a numeric value between, and including, 0 to 1, because it is a percentile value. The return value is always in the set of values of `<expression>`.

Examples

```sql
CREATE ROW TABLE p_disc (class CHAR(10), val INT, offset INT);
  INSERT INTO p_disc VALUES('A', 1, 1);
  INSERT INTO p_disc VALUES('A', 3, 3);
  INSERT INTO p_disc VALUES('A', 5, null);
  INSERT INTO p_disc VALUES('A', 5, 2);
  INSERT INTO p_disc VALUES('A', 10, 0);
  INSERT INTO p_disc VALUES('B', 1, 3);
  INSERT INTO p_disc VALUES('B', 1, 1);
  INSERT INTO p_disc VALUES('B', 7, 1);

SELECT class,
     val,
     PERCENTILE_DISC(0.125) WITHIN GROUP (ORDER BY val) OVER (PARTITION BY class) AS pd1,
     PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY val) OVER (PARTITION BY class) AS pd2,
     PERCENTILE_DISC(0.875) WITHIN GROUP (ORDER BY val) OVER (PARTITION BY class) AS pd3
FROM p_disc;
```

<table>
<thead>
<tr>
<th>CLASS</th>
<th>VAL</th>
<th>PD1</th>
<th>PD2</th>
<th>PD3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
4.9.1.150 PLAINTEXT Function (Fulltext)

Returns the plain text representation of a value in a column that has a full text index.

Syntax

```
PLAINTEXT(<column_name>)
```

Syntax Elements

- `column_name`
  
  Specifies the identifier for a column that has a full text index on it.

Description

The PLAINTEXT function returns the plain text representation of a value in a column that has a full text index. Use this function when you have binary data, such as a PDF, as a column value.

If the specified column does not have a full text index on it, then an error is returned.

Example

The following example runs the PLAINTEXT function on a fictitious column value `<somePDF>` (a PDF) and returns the plain text representation of the value. This is an example;<somePDF>:

```
CREATE COLUMN TABLE T (CONTENT BLOB);
CREATE FULLTEXT INDEX I ON T(CONTENT);
INSERT INTO T VALUES (<somePDF>);
SELECT PLAINTEXT(CONTENT), CONTENT FROM T;
```
4.9.1.151 POWER Function (Numeric)

Calculates a specified base number raised to the power of a specified exponent.

Syntax

```
POWER(<base>, <exponent>)
```

Description

Calculates a specified base number raised to the power of a specified exponent.

Example

The following example returns the value 1024:

```
SELECT POWER (2, 10) "power" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.152 QUARTER Function (Datetime)

Returns the quarter for the specified date.

Syntax

```
QUARTER(<date>, [, <start_month> ])
```
Description

Returns the numerical year and quarter of date <date>. The first quarter starts in the month specified by <start_month>. If <start_month> is not specified, then the first quarter is assumed to begin in January.

Example

The following example returns the value 2011–Q4 as the year and quarter for the specified date:

```
SELECT QUARTER (TO_DATE('2012-01-01', 'YYYY-MM-DD'), 2) "quarter" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]

4.9.1.153 RAND Function (Numeric)

Returns a pseudo-random DOUBLE value.

Syntax

```
RAND()
```

Description

Returns a pseudo-random DOUBLE value in the range of 0 to less than 1.0.

Values generated by the RAND function are not safe for cryptographic or security purposes. Use the RAND_SECURE function for values that are safe for cryptographic or security purposes.

Example

The following example returns a pseudo-random DOUBLE value similar to 0.4610119133779396:

```
SELECT RAND() FROM DUMMY;
```
4.9.1.154 RANDOM_PARTITION Function (Window)

Partitions an input set randomly into three disjoint subsets by assigning a set number to each row of the input set.

Syntax

```
RANDOM_PARTITION(<training_set_size>, <validation_set_size>, <test_set_size>, <seed>) <window_specification>
```

Syntax Elements

- `window_specification`
  
  Defines a window on the data over which the function operates. For `window_specification`, see Window Functions and the Window Specification [page 424].

Description

The three subsets that RANDOM_PARTITION partitions the set into are called training, validation, and testing. The union of these three subsets may not be the complete initial dataset.

- `<training_set_size>`, `<validation_set_size>`, and `<test_set_size>` can be numeric values that specify the sizes of the training, validation, and testing sets, respectively. When all three values are less than one, the sizes are expressed as fractions of the total input size. When all three values are greater than or equal to one, they represent the actual number of elements in the different partitions. `<seed>` specifies the seed for the random number generation. When set to zero, the `<seed>` is initialized with the current time.

The set numbers are defined as follows:

<table>
<thead>
<tr>
<th>Set number</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The value is not assigned to a set</td>
</tr>
</tbody>
</table>
Stratified partitions are supported via the PARTITION BY clause.

To generate reproducible partitions by providing a value for the `<seed>`, ensure that the input set is ordered in the same way in each call to the function. The `<seed>` value must be larger than zero to generate reproducible partitions; a value of zero does not create reproducible partitions. Do not use `<seed>` without the ORDER BY clause. The partitions can be non-deterministic among tie values in the `<window_order_by_expression>`.

**Examples**

**Stratified partitions**

The following example creates stratified partitions (random partitions per station with fractional partition sizes). Since the partition numbers are randomly assigned to the rows, the result might differ from call to call.

```
CREATE TABLE weather (station INT, ts DATE, temperature FLOAT);
INSERT INTO weather VALUES (1, '2014-01-01', 0);
INSERT INTO weather VALUES (1, '2014-01-02', 3);
INSERT INTO weather VALUES (1, '2014-01-03', 4.5);
INSERT INTO weather VALUES (1, '2014-01-04', 6);
INSERT INTO weather VALUES (1, '2014-01-05', 6.3);
INSERT INTO weather VALUES (1, '2014-01-06', 5.9);
INSERT INTO weather VALUES (2, '2014-01-01', 1);
INSERT INTO weather VALUES (2, '2014-01-02', 3.4);
INSERT INTO weather VALUES (2, '2014-01-03', 5);
INSERT INTO weather VALUES (2, '2014-01-04', 6.7);
INSERT INTO weather VALUES (2, '2014-01-05', 4.6);
INSERT INTO weather VALUES (2, '2014-01-06', 6.9);
SELECT *, RANDOM_PARTITION(0.5, 0.2, 0.3, 0) OVER (PARTITION BY station ORDER BY ts) AS part_num FROM weather;
```

<table>
<thead>
<tr>
<th>STATION</th>
<th>TS</th>
<th>TEMPERATURE</th>
<th>PART_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01.01.2014</td>
<td>0.0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>02.01.2014</td>
<td>3.0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>03.01.2014</td>
<td>4.5</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>04.01.2014</td>
<td>6.0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>05.01.2014</td>
<td>6.3</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>06.01.2014</td>
<td>5.9</td>
<td>3</td>
</tr>
</tbody>
</table>
### Random partitions with explicit partition sizes

The following example creates random partitions with explicit partition sizes:

```sql
DROP TABLE weather;
CREATE ROW TABLE weather (station INT, ts DATE, temperature FLOAT);
INSERT INTO weather VALUES(1, '2014-01-01', 0.0);
INSERT INTO weather VALUES(1, '2014-01-02', 3.0);
INSERT INTO weather VALUES(1, '2014-01-03', 4.5);
INSERT INTO weather VALUES(1, '2014-01-04', 6.0);
INSERT INTO weather VALUES(1, '2014-01-05', 6.3);
INSERT INTO weather VALUES(1, '2014-01-06', 5.9);
INSERT INTO weather VALUES(2, '2014-01-01', 1.0);
INSERT INTO weather VALUES(2, '2014-01-02', 3.4);
INSERT INTO weather VALUES(2, '2014-01-03', 5.0);
INSERT INTO weather VALUES(2, '2014-01-04', 6.7);
INSERT INTO weather VALUES(2, '2014-01-05', 4.6);
INSERT INTO weather VALUES(2, '2014-01-06', 6.9);
SELECT *, RANDOM_PARTITION(5, 2, 3, 0) OVER (ORDER BY temperature) AS part_num
FROM weather;
```

<table>
<thead>
<tr>
<th>STATION</th>
<th>TS</th>
<th>TEMPERATURE</th>
<th>PART_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01.01.2014</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>01.01.2014</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>02.01.2014</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>02.01.2014</td>
<td>3.4</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>03.01.2014</td>
<td>4.5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>05.01.2014</td>
<td>4.6</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>03.01.2014</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>06.01.2014</td>
<td>5.9</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>04.01.2014</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>05.01.2014</td>
<td>6.3</td>
<td>3</td>
</tr>
</tbody>
</table>
### 4.9.1.155 RAND_SECURE Function (Numeric)

Returns a pseudo-random value that is safe for cryptographic or security purposes.

#### Syntax

```
RAND_SECURE()
```

#### Description

Returns a DOUBLE value in the range of 0.0 to less than 1.0.

Values generated by the RAND_SECURE function are safe for cryptographic or security purposes.

The RAND function offers better performance than the RAND_SECURE function, but returns a value that is not safe for cryptographic or security purposes.

#### Example

The following example returns a DOUBLE value similar to 0.9646101191337793.

```
SELECT RAND_SECURE() from dummy;
```
4.9.1.156  RANK Function (Window)

Returns rank of a row within a partition, starting from 1.

Syntax

RANK() <window_specification>

Syntax Elements

window_specification

Defines a window on the data over which the function operates. For <window_specification>, see Window Functions and the Window Specification [page 424].

Description

The RANK function returns duplicate values in the ranking sequence when there are ties between values and the next rankings are skipped.

Examples

In this example, the data is partitioned by the product name, and the sales of types within the products are ranked. The plain ballcap is ranked 4 instead of 3 because of the sales numbers for the team logo and lettered versions of the ballcap, which are tied at rank 2.

CREATE ROW TABLE ProductSales(ProdName VARCHAR(50), Type VARCHAR(20), Sales INT);
INSERT INTO ProductSales VALUES('Tee Shirt','Plain',21);
INSERT INTO ProductSales VALUES('Tee Shirt','Lettered',22);
INSERT INTO ProductSales VALUES('Tee Shirt','Team logo',30);
INSERT INTO ProductSales VALUES('Hoodie','Plain',60);
INSERT INTO ProductSales VALUES('Hoodie','Lettered',65);
INSERT INTO ProductSales VALUES('Hoodie','Team logo',80);
INSERT INTO ProductSales VALUES('Ballcap','Vintage',60);
INSERT INTO ProductSales VALUES('Ballcap','Plain',8);
INSERT INTO ProductSales VALUES('Ballcap','Lettered',40);
INSERT INTO ProductSales VALUES('Ballcap','Team logo',40);
SELECT ProdName, Type, Sales,
RANK() OVER ( PARTITION BY ProdName ORDER BY Sales DESC ) AS Rank
FROM ProductSales
ORDER BY ProdName, Type;

<table>
<thead>
<tr>
<th>PRODNAME</th>
<th>Type</th>
<th>SALES</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ballcap</td>
<td>Lettered</td>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>Ballcap</td>
<td>Plain</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Ballcap</td>
<td>Team logo</td>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>Ballcap</td>
<td>Vintage</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Lettered</td>
<td>65</td>
<td>2</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Plain</td>
<td>60</td>
<td>3</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Team logo</td>
<td>80</td>
<td>1</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Lettered</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Plain</td>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Team logo</td>
<td>30</td>
<td>1</td>
</tr>
</tbody>
</table>

Related Information

Expressions [page 56]
Window Functions and the Window Specification [page 424]

4.9.1.157 RECORD_COMMIT_TIMESTAMP Function (Miscellaneous)

Returns a COMMIT timestamp for the specified row of the given table.

Syntax

RECORD_COMMIT_TIMESTAMP(<table_name_or_alias>)
Syntax Elements

table-name-or-alias Specifies the name or alias of the table to return the COMMIT timestamps for.

Description

Returns a COMMIT timestamp for the specified row of the given table if the table is tracking COMMIT timestamps for each row. The COMMIT timestamp is based on an internal counter that increases on commit operations and does not contain a date or time. If the table is not tracking COMMIT timestamps, the function returns NULL.

Depending on the table type, DDL statements such as ALTER TABLE can affect the value returned by the RECORD_COMMIT_TIMESTAMP function:

• For column store tables, the returned value is not affected by DDL operations.
• For row store tables, the returned value is changed to the COMMIT timestamp of the transaction that performed the DDL operation.

Example

Execute the following statements to create and populate two tables, AA and BB, where AA records COMMIT timestamps and BB does not.

```sql
CREATE TABLE AA (A int) RECORD COMMIT TIMESTAMP;
CREATE TABLE BB (B int);
INSERT INTO AA VALUES (1);
INSERT INTO BB VALUES (1);
COMMIT;
```

Assume the transaction is committed with a COMMIT timestamp of 10. Executing the following statement returns the COMMIT timestamp of the transaction, in this case, 10.

```sql
SELECT RECORD_COMMIT_TIMESTAMP(AA) FROM AA;
```

Executing a similar statement on table BB returns NULL since the RECORD COMMIT TIMESTAMP option was not set for table BB.

```sql
SELECT RECORD_COMMIT_TIMESTAMP(BB) FROM BB;
```
4.9.1.158 RECORD_ID Function (Miscellaneous)

Generates an ID for each row of the results.

Syntax

```
RECORD_ID( <table> )
```

Syntax Elements

`table`

Specifies the column store or row store table to generate row IDs for.

`<t>` should be located in the FROM clause that is syntactically closest, and cannot be a temporary table, derived table, or any other type of subquery. For example, these syntaxes are supported:

```
SELECT RECORD_ID(t) FROM t;
SELECT RECORD_ID(tab) FROM t AS tab;
```

However, these syntaxes are not supported:

```
SELECT RECORD_ID(v) FROM (SELECT * FROM t) AS v;
SELECT a, RECORD_ID(st) FROM (SELECT * FROM t) AS st;
```

Description

Returns a result set with a unique BIGINT value for each row of the results. These values remain valid for the current transaction.

Examples

The following example shows the use of the RECORD_ID function to select a value from a specified row ID:

```
CREATE COLUMN TABLE T1 (column1 INT);
INSERT INTO T1 VALUES (5);
INSERT INTO T1 VALUES (10);
SELECT RECORD_ID(T1) AS RECORD_ID FROM T1;
```
The SELECT query returns the following result:

<table>
<thead>
<tr>
<th>RECORD_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

This SELECT query returns the following result:

```
SELECT * FROM T1 WHERE RECORD_ID(T1) = 2;
```

<table>
<thead>
<tr>
<th>COLUMN1</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

### 4.9.1.159 REPLACE Function (String)

Searches within a string for all occurrences of a specified string and replaces them with another specified string.

#### Syntax

```
REPLACE(<original_string>, <search_string>, <replace_string>)
```

#### Syntax Elements

- `original_string`
  
  Specifies the string to perform the search and replace within. If `<original_string>` is empty, then the result is an empty string.

- `search_string`
  
  Specifies the pattern to search for in `<original_string>`.  

- `replace_string`
  
  Specifies the string to replace `<search_string>` with.
**Description**

Searches in `<original_string>` for all occurrences of `<search_string>` and replaces them with `<replace_string>`.

- If two overlapping substrings match the `<search_string>` in the `<original_string>`, then only the first occurrence is replaced.
- If `<original_string>` does not contain an occurrence of `<search_string>`, then `<original_string>` is returned unchanged.
- If `<original_string>`, `<search_string>`, or `<replace_string>` are NULL, then NULL is returned.

**Example**

The following example changes all occurrences of `DOWN` in the original string to `UP`, and returns the value `UPGRADE UPWARD`:

```sql
SELECT REPLACE ('DOWNGRADE DOWNWARD','DOWN', 'UP') "replace" FROM DUMMY;
```

**Related Information**

Character String Data Types [page 36]

4.9.1.160 REPLACE_REGEXPR Function (String)

Searches a string for a regular expression pattern and returns the string with either one or every occurrence of the regular expression pattern that is replaced using a replacement string.

**Syntax**

```
REPLACE_REGEXPR(\<pattern\> [ \text{FLAG} \text{flag} ]
\IN \<regex\_subject\_string\>
[ \text{WITH} \<replacement\_string\> ]
[ \text{FROM} \<start\_position\> ]
[ \text{OCCURRENCE} \<regex\_replace\_occurrence\> ])
```

**Syntax Elements**

- `pattern`
Specifies a search pattern based on Perl Compatible Regular Expression (PCRE).

\[ \text{<pattern>} ::= \text{!!Perl Compatible Regular Expression} \]

**Flag**

Specifies the matching behavior to use.

\[ \text{<flag>} ::= 'i' \mid 'm' \mid 's' \mid 'x' \]

**Flag options**

<table>
<thead>
<tr>
<th>Flag option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Enables case-insensitive matching</td>
</tr>
<tr>
<td>m</td>
<td>Enables multiline mode, where the (&lt;\text{subject_string}&gt;) will be treated as multiple lines and the expression (^\text{^}) and (^\text{$}) match just after or just before, respectively, a line terminator or the end of the input sequence</td>
</tr>
<tr>
<td>s</td>
<td>Enables the expression (&lt;.&gt;) as a wildcard to match any character, including a line terminator</td>
</tr>
<tr>
<td>x</td>
<td>Permits whitespace and comments in the pattern</td>
</tr>
</tbody>
</table>

**U**

SAP HANA uses the Perl-compatible Regular Expressions (PCRE) library to process regular expressions. Specifying 'U' (short for "ungreedy") inverts the "greediness" of quantifiers so that they are not greedy by default but become greedy only when followed by '?' . Ungreedy matching can often perform faster because it finds the shorter match at times when it is only interesting to know whether there is any match.

For a full understanding of the "greedy" versus "ungreedy" matching behavior of the Perl-compatible Regular Expressions (PCRE) library, visit: https://www.pcre.org/original/doc/html/pcre-matching.html .

**regex_subject_string**

Specifies the string that the search pattern should be applied to. If \(<\text{regex_subject_string}>\) is empty, the result is empty.

\[ \text{<regex_subject_string>} ::= \text{<string_literal>} \]

**replacement_string**

Specifies the string that replaces all occurrences found by the function. The default is an empty string.

\[ \text{<replacement_string>} ::= \text{<string_literal>} \]

**start_position**

Specifies the character of \(<\text{regex_subject_string}>\) where the search is started, if the value is a positive integer. Otherwise, NULL is returned.

\[ \text{<start_position>} ::= \text{<numeric_literal>} \]

**regex_replace_occurrence**
Specifies a non-negative integer or ALL, and indicates the occurrence of the replace operation. If `<regex_replace_occurrence>` is a negative integer, then the `<regex_subject_string>` is returned without any change. The default value is ALL.

\[<regex_replace_occurrence> ::= <numeric_literal> | ALL\]

**Description**

Searches a string for a regular expression pattern and returns the string with either one or every occurrence of the regular expression pattern that is replaced using a replacement string.

If any of the following parameters is NULL, then the function returns NULL: `<pattern>`, `<flag>`, `<regex_subject_string>`, `<replacement_string>`, `<start_position>` or `<regex_replace_occurrence>`.

**Example**

The following example uses a regular expression to replace date format `2014-04-01` with the value `01/04/2014`:

```
SELECT REPLACE_REGEXPR('([[:digit:]][{4}])-(\[[[:digit:]][2]\])-(\[[[:digit:]][2]\])' IN '2014-04-01' WITH '\3/\2/\1' OCCURRENCE ALL) "replace_regexpr" FROM DUMMY;
```

**Related Information**

Character String Data Types [page 36]

4.9.1.161 RESULT_CACHE_ID Function (Miscellaneous)

Returns the cache ID of a result cache entry.

**Syntax**

```
RESULT_CACHE_ID()
```
Description

If no result cache entry is specified, then NULL is returned.

Example

The following fictitious example returns the cache ID for result cache entry V1, 40000001.

```sql
SELECT DISTINCT RESULT_CACHE_ID () FROM V1 WITH HINT(RESULT_CACHE);
```

4.9.1.162 RESULT_CACHE_REFRESH_TIME Function (Miscellaneous)

Returns the last cache refresh time of a result cache entry.

Syntax

```
RESULT_CACHE_REFRESH_TIME()
```

Description

If no result cache entry is provided, then NULL is returned.

Example

The following fictitious example returns the last cache refresh time of the result cache entry V1, 2015-11-20 01:01:01.010101.

```sql
SELECT DISTINCT RESULT_CACHE_REFRESH_TIME() FROM V1;
```
4.9.1.163  RIGHT Function (String)

Returns the specified number of characters/bytes of a string, starting from the right side.

Syntax

RIGHT(<string>, <number>)

Syntax Elements

string
Specifies the string to be operated on.

number
Specifies the length of characters or bytes to return, starting from the right side.

Description

Returns the rightmost <number> characters/bytes of string <string>.

Returns an empty string value if <number> is less than 1.

Returns string specified by <string> (without blank padding) if the value of <number> is greater than the length of string specified by <string>.

Example

The following example returns the rightmost 3 characters of the given string (789):

```
SELECT RIGHT('HI0123456789', 3) "right" FROM DUMMY;
```

The following example returns the entire input string because the value 20 exceeds the length of the input string:

```
SELECT RIGHT('HI0123456789', 20) "right" FROM DUMMY;
```
4.9.1.164  ROUND Function (Numeric)

Rounds the specified argument to the specified amount of places after the decimal point.

Syntax

\[
\text{ROUND}(\text{<number>}, \text{<position>}, \text{<rounding_mode>})
\]

Syntax Elements

| number | Specifies the numeric argument to round. |
| position | Specifies the amount of places after the decimal point to round <number> to. The default value is 0. A positive or negative integer rounds to the specified position to the right or left of the decimal point, respectively. |
| rounding_mode | Defines how the rounding should be carried out. |

| <rounding_mode> ::= | ROUND_HALF_UP | ROUND_HALF_DOWN | ROUND_HALF_EVEN | ROUND_UP | ROUND_DOWN | ROUND_CEILING | ROUND_FLOOR |

| ROUND_HALF_UP | The value is rounded up to the next round figure. This is the default. If the value falls precisely halfway between two rounded values, it is rounded up away from zero (as is done in commercial rounding). |

| ROUND_HALF_DOWN | The value is rounded down to the next round figure. If the value falls precisely halfway between two round values, it is rounded down towards zero. |

| ROUND_HALF_EVEN | |
The value is rounded to the next round figure. If the value falls precisely halfway between two rounded values, it is rounded to the value whose last decimal place is an even number.

**ROUND_UP**

The value is always rounded away from zero, to the larger round figure.

**ROUND_DOWN**

The value is always rounded towards zero, to the smaller round figure.

**ROUND_CEILING**

The value is always rounded in a positive direction, to the larger value.

**ROUND_FLOOR**

The value is always rounded in a negative direction, to the smaller value.

**Description**

Rounds argument `<n>` to the specified amount of places (`<pos>`) after the decimal point.

When using ROUND with floating point types, REAL and DOUBLE, the precision of the numeric representation can affect the result. In this case, the actual number of digits after the decimal point is influenced by the precision of the type used. For example, the result of the following statement is `399.7099914550781` not `399.71`:

```sql
SELECT ROUND(TO_REAL(399.71429443359375),2) from DUMMY;
```

**Examples**

The following example returns the value `16.2`:

```sql
SELECT ROUND (16.16, 1) "round" FROM DUMMY;
```

The following example returns the value `20`:

```sql
SELECT ROUND (16.16, -1) "round" FROM DUMMY;
```

The following example returns the value `438.8`:

```sql
SELECT ROUND( 438.75, 1, ROUND_HALF_UP) "round" FROM DUMMY;
```

The following example returns the value `438.7`:

```sql
SELECT ROUND( 438.75, 1, ROUND_HALF_DOWN) "round" FROM DUMMY;
```

The following example returns the value `438.8`:

```sql
SELECT ROUND( 438.75, 1, ROUND_HALF_EVEN) "round" FROM DUMMY;
```
The following example returns the value 438.8:

```
SELECT ROUND( 438.71, 1, ROUND_UP) "round" FROM DUMMY;
```

The following example returns the value 438.7:

```
SELECT ROUND( 438.79, 1, ROUND_DOWN) "round" FROM DUMMY;
```

The following example returns the value 438.8:

```
SELECT ROUND( 438.75, 1, ROUND_CEILING) "round" FROM DUMMY;
```

The following example shows the difference when specifying a positive and negative integer for `<pos>`:

```
SELECT ROUND(1234.1234, 1)"round" FROM DUMMY;  ==> 1234.1000
SELECT ROUND(1234.1234, -1)"round" FROM DUMMY; ==> 1230.0000
```

**Related Information**

Numeric Data Types [page 48]

### 4.9.1.165 RPAD Function (String)

Right-pads a string with spaces or a specified pattern to make a string that is a specified number of characters in length.

**Syntax**

```
RPAD(<string>, <number> [, <pattern>])
```

**Syntax Elements**

- **string**
  Specifies the string to be padded.

- **number**
  Specifies the length to pad `<string>`. `<number>` must be an integer.

- **pattern**
  Specifies a string of characters, rather than spaces, to use for padding.
Description

Right-pads the end of `<string>` with spaces or characters to make a string of `<number>` characters in length. If `<pattern>` is specified, then `<string>` is padded using sequences of the given characters until the required length is met.

If the length of `<string>` is greater than `<n>`, then no padding is performed and the resulting value is truncated to the length specified in `<number>`.

RPAD returns an empty string value if `<number>` is less than 1.

Example

The following example right-pads the end of string `end` with the pattern `12345` to make a string of `15` characters in length and returns the value `end123451234512`:

```
SELECT RPAD ('end', 15, '12345') "right padded" FROM DUMMY;
```

In the following example, `<str>` is longer than `<n>`, so no padding is performed and the result is `<str>` truncated to the length of `<n>` (that is, `en`):

```
SELECT RPAD ('end', 2, '12345') "right padded" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]

4.9.1.166  RTRIM Function (String)

Returns a string trimmed of all trailing spaces.

Syntax

```
RTRIM(<string> [,<remove_set> ])
```
**Description**

Returns a string trimmed of all trailing spaces. If `<remove_set>` is specified, then `RTRIM` removes all the characters contained in the specified set from the end of the string. This process continues until a character that is not in the `<remove_set>` has been reached.

 `<remove_set>` is treated as a set of characters and not as a search string.

**Example**

This example removes all trailing `a` and `b` characters from the given string, and returns the value `endabA`:

```sql
SELECT RTRIM ('endabAabbabab','ab') "rtrim" FROM DUMMY;
```

**Related Information**

- Character String Data Types [page 36]

### 4.9.1.167  ROW_NUMBER Function (Window)

Sequentially numbers the rows within a partition of a result set, with the first row of each partition assigned as 1.

**Syntax**

```
ROW_NUMBER() <window_specification>
```

**Syntax Elements**

- `window_specification`  
  Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].
Description

The output of the ROW_NUMBER function can be non-deterministic among tie values. The ordering of the sequence is determined by the `<windows_order_by_clause>` within the OVER windows clause.

Examples

The following example returns the sales figures sorted in descending number by product type. The data is partitioned by the product name, and the row number is returned for each type within its product.

```
CREATE ROW TABLE ProductSales(ProdName VARCHAR(50), Type VARCHAR(20), Sales INT);
INSERT INTO ProductSales VALUES('Tee Shirt','Plain',21);
INSERT INTO ProductSales VALUES ('Tee Shirt','Lettered',22);
INSERT INTO ProductSales VALUES ('Tee Shirt','Team logo',30);
INSERT INTO ProductSales VALUES('Hoodie','Plain',60);
INSERT INTO ProductSales VALUES ('Hoodie','Lettered',65);
INSERT INTO ProductSales VALUES ('Hoodie','Team logo',80);
INSERT INTO ProductSales VALUES('Ballcap','Plain',8);
INSERT INTO ProductSales VALUES ('Ballcap','Lettered',40);
INSERT INTO ProductSales VALUES ('Ballcap','Team logo',27);
SELECT ProdName, Type, Sales,
    ROW_NUMBER() OVER (PARTITION BY ProdName ORDER BY Sales DESC) AS row_num
FROM ProductSales
ORDER BY ProdName, Sales DESC;
```

<table>
<thead>
<tr>
<th>PRODNAME</th>
<th>DESCRIPTION</th>
<th>SALES</th>
<th>ROW_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ballcap</td>
<td>Lettered</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>Ballcap</td>
<td>Team logo</td>
<td>27</td>
<td>2</td>
</tr>
<tr>
<td>Ballcap</td>
<td>Plain</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Team logo</td>
<td>80</td>
<td>1</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Lettered</td>
<td>65</td>
<td>2</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Plain</td>
<td>60</td>
<td>3</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Team logo</td>
<td>30</td>
<td>1</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Lettered</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Plain</td>
<td>21</td>
<td>3</td>
</tr>
</tbody>
</table>

In the following example, the ROW_NUM values for the items are different because there is no `<windows_order_by_clause>` specification:

```
SELECT ProdName, Type, Sales,
    ROW_NUMBER() OVER (PARTITION BY ProdName) AS row_num
FROM ProductSales
ORDER BY ProdName, Sales DESC;
```
<table>
<thead>
<tr>
<th>PRODNAME</th>
<th>DESCRIPTION</th>
<th>SALES</th>
<th>ROW_NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ballcap</td>
<td>Lettered</td>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>Ballcap</td>
<td>Team logo</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>Ballcap</td>
<td>Plain</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Team logo</td>
<td>80</td>
<td>3</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Lettered</td>
<td>65</td>
<td>2</td>
</tr>
<tr>
<td>Hoodie</td>
<td>Plain</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Team logo</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Lettered</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>Tee Shirt</td>
<td>Plain</td>
<td>21</td>
<td>1</td>
</tr>
</tbody>
</table>

**Related Information**

*Expressions [page 56]*
*Window Functions and the Window Specification [page 424]*

### 4.9.1.168 SCORE Function (Fulltext)

Returns the relevance of a record that has been found.

**Syntax**

**Syntax 1 - Without Parameters**

```sql
SCORE()
```

**Syntax 2 - With Parameters**

```sql
SCORE ( <search_string> IN <score_columns> )
```
Syntax Elements

score_columns

<score_columns> ::= { <single_column> | <specific_columns_diff_options> | <specific_columns_same_options> | <all_columns> }

single_column

<single_column> ::= <column_name> [ <scoring_type> ]

scoring_type

<scoring_type> ::= { EXACT [ <exact_options> ] | FUZZY [ <fuzzy_options> ] | LINEAR <linear_scoring_details> | GAUSSIAN <gaussian_scoring_details> | LOGARITHMIC <logarithmic_scoring_details> }

exact_options

<exact_options> ::= { ABBREVIATION SIMILARITY <value_range> | AND SYMMETRIC {'ON' | 'OFF'} | AND THRESHOLD <value_range> | MINIMAL SCORE <value_range> | WEIGHT <value_range_2> | COMPOUND WORD WEIGHT <value_range_2> | NON MATCHING TOKEN MODE {'max' | 'min' | 'all' | 'input' | 'table'} | EXCESS TOKEN WEIGHT <value_range> | PHRASE CHECK FACTOR <value_range> | SEARCH MODE <search_mode> }

fuzzy_search

<fuzzy_options> ::= { ABBREVIATION SIMILARITY <value_range> | AND SYMMETRIC {'ON' | 'OFF'} | AND THRESHOLD <value_range> | MINIMAL SCORE <value_range> | WEIGHT <value_range_2> | COMPOUND WORD WEIGHT <value_range_2> | NON MATCHING TOKEN MODE {'max' | 'min' | 'all' | 'input' | 'table'} | EXCESS TOKEN WEIGHT <value_range> | SUBSTRING MATCH {'off' | 'anywhere' | 'beginning'} | LANGUAGE TRANSLATION {'on' | 'off'} | LENGTH TOLERANCE <value_range> | MAX DATE DISTANCE { 0 | <positive_integer_constant> } | MINIMAL SEARCH LENGTH { 0 | <positive_integer_constant> } | PHRASE CHECK FACTOR <value_range> | SEARCH MODE <search_mode> | SIMILARITY CALCULATION MODE <similar_calculation_mode> | SPELL CHECK FACTOR <value_range> | MINIMAL TOKEN SCORE <value_range> }

value_range_1 A value that is greater than or equal to 0.0 and less than or equal to 1. For example, 0.326, 0.57, or 0.9.
**value_range_2** A value that is greater than 0 and less than 1.0.

**positive_float_constant** A positive integer greater than 1.0. For example, 57.326.

**positive_integer_constant** A positive integer constant. For example, 5.

**float_constant** A non-negative value.

**search_mode**

```plaintext
<search_mode> ::= 
  { 'alphanum'
    | 'alphanum identifier'
    | 'house number'
    | 'identifier'
    | 'postcode'
    | 'string'
    | 'text')
```

**similar_calculation_mode**

```plaintext
<similar_calculation_mode> ::= 
  { 'compare'
    | 'type ahead'
    | 'search compare'
    | 'search'
    | 'symmetric search'
    | 'substring search'
    | 'flexible')
```

**linear_scoring_details**

```plaintext
<scoring_details> ::= 
  [ MINIMAL SCORE <value_range> ]
  [ OFFSET { 0 | <positive_float_constant> ]
  [ DECAY <linear_decay_value> ]
  [ SCALE <positive_float_constant> ]
  [ SEARCH MODE { 'date' | 'numc' }]
  [ WEIGHT <weight_value> ]
```

**gaussian_scoring_details**

```plaintext
<scoring_details> ::= 
  [ MINIMAL SCORE <value_range> ]
  [ OFFSET { 0 | <positive_float_constant> ]
  [ DECAY <gaussian_decay_value> ]
  [ SCALE <positive_float_constant> ]
  [ SEARCH MODE { 'date' | 'numc' }]
  [ WEIGHT <weight_value> ]
```

**linear_decay_value** A value greater than or equal to 0.0 but less than 1.0.

**gaussian_decay_value** A value greater than 0.0 but less than 1.0.

**weight_value** A value greater than 0.0 and less than or equal to 1.0.

**logarithmic_scoring_details**

```plaintext
<logarithmic_scoring_details> ::= 
  [ MINIMAL SCORE <value_range> ]
  [ OFFSET { 0 | <positive_float_constant> ]
  [ BASE <float_constant> ]
  [ SEARCH MODE 'numc' ]
  [ WEIGHT <weight_value> ]
```

**specific_columns_diff_options**

```plaintext
<specific_columns_diff_options> ::= 
  ( <column_name> [ , <column_name> ],... )
```
specific_columns_same_options

<specific_columns_same_options> ::= ( <column_name> [, <column_name> ],... ] ) [ <scoring_type> ]

all_columns

<all_columns> ::= * [ <scoring_type> ]

Description

Syntax 1 can only be used in search queries using the CONTAINS predicate. Use SCORE to obtain the relevance of a record that has been found. SCORE returns a real value between 0 and 1. The SAP HANA database calculates a score based on the following information:

- The relevance, or weighting, of attributes in a search using the CONTAINS predicate. The relevance of a match depends on the weight of the column that caused the match. You can specify weights as you create the view, or in the CONTAINS predicate.
- Fuzziness in a fuzzy search. The more exact matching that occurs, the higher the score is.
- Text ranking (TF-IDF).

Syntax 2 cannot be used with the CONTAINS predicate. It replaces Syntax 1.

Examples

Syntax 1

The following example creates a table that contains two strings and then inserts values into the table.

```
CREATE TABLE T1 (CONTENT TEXT FAST PREPROCESS OFF);

INSERT INTO T1 VALUES('This is a test.');  INSERT INTO T1 VALUES('This was a test.');
```

Use the SCORE function to check the table contents for relevance against the search string 'is':

```
SELECT SCORE(), CONTENT FROM T1 WHERE CONTAINS(CONTENT, 'is', LINGUISTIC);
```

<table>
<thead>
<tr>
<th>SCORE</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8700000047683716</td>
<td>This is a test.</td>
</tr>
<tr>
<td>0.4083336114883423</td>
<td>This was a test.</td>
</tr>
</tbody>
</table>

Still using TABLE T1, the following example adds additional values to the table.

```
INSERT INTO T1 VALUES('example');
```

INSERT INTO T1 VALUES('example');

Use the SCORE function to check the table contents for similarity to the string 'example':

```
SELECT SCORE(), CONTENT FROM T1 WHERE CONTAINS(CONTENT, 'example', Fuzzy(0.8));
```

<table>
<thead>
<tr>
<th>SCORE</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>example</td>
</tr>
<tr>
<td>1</td>
<td>example</td>
</tr>
</tbody>
</table>

**Syntax 2**

The following example creates a table that contains two strings:

```
CREATE TABLE T2 (CONTENT TEXT FAST PREPROCESS OFF);
```

```
INSERT INTO T2 VALUES('This is a test.');
INSERT INTO T2 VALUES('This was a test.');
```

Use the SCORE function to check the table contents for relevance against the search string 'is':

```
SELECT SCORE('is' IN CONTENT) FROM T2 WHERE SCORE('is' IN CONTENT) > 0.2;
```

<table>
<thead>
<tr>
<th>SCORE</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8700000047683716</td>
<td>This is a test.</td>
</tr>
<tr>
<td>0.40833336114883423</td>
<td>This was a test.</td>
</tr>
</tbody>
</table>

Still using TABLE T2, the following example adds additional values to the table.

```
INSERT INTO T2 VALUES('example');
INSERT INTO T2 VALUES('example');
```

Use the SCORE function to check the table contents for similarity to the string 'example':

```
SELECT SCORE('example' IN CONTENT, FUZZY MINIMAL TOKEN SCORE 0.8) FROM T2 WHERE SCORE('example' IN CONTENT, FUZZY MINIMAL TOKEN SCORE 0.8) > 0.8;
```

<table>
<thead>
<tr>
<th>SCORE</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>example</td>
</tr>
<tr>
<td>1</td>
<td>example</td>
</tr>
</tbody>
</table>
4.9.1.169  SECOND Function (Datetime)

Returns the second portion of a specified time.

Syntax

```
SECOND(<time>)
```

Description

Returns a value of seconds for a given time.
Subseconds are included for TIMESTAMP datatypes.

Examples

The following example returns the value 56 for the seconds of the specified time:

```
SELECT SECOND ('12:34:56') "second" FROM DUMMY;
```

The following example returns the value 56.789 for the subseconds of the specified time:

```
SELECT SECOND ('2014-03-25 12:34:56.789') "subseconds" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]
4.9.1.170 SECONDS_BETWEEN Function (Datetime)

Computes the number of seconds between two specified dates.

Syntax

```sql
SECONDS_BETWEEN(<date_1>, <date_2>)
```

Description

Computes the number of seconds between date arguments `<date_1>` and `<date_2>`, which is semantically equal to (`<date_2>` - `<date_1>`).

Example

The following example returns the value 2,678,400 as the number of seconds between the two specified dates:

```sql
SELECT SECONDS_BETWEEN ('2009-12-05', '2010-01-05') "seconds between" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]

4.9.1.171 SERIES_DISAGGREGATE Function (Series Data)

Generates a complete series table with rows disaggregated into defined partitions.

Syntax

```sql
SERIES_DISAGGREGATE(
    {SERIES TABLE <source_series_table> | <source_increment_by>},
    {SERIES TABLE <generate_series_table> | <target_increment_by>}
    [, <value_range> ]
)
```
{ SERIES_DISAGGREGATE_TINYINT
SERIES_DISAGGREGATE_SMALLINT
SERIES_DISAGGREGATE_INTEGER
SERIES_DISAGGREGATE_BIGINT
SERIES_DISAGGREGATE_SMALLDECIMAL
SERIES_DISAGGREGATE_DECIMAL
SERIES_DISAGGREGATE_TIME
SERIES_DISAGGREGATE_DATE
SERIES_DISAGGREGATE_SECONDDATE
SERIES_DISAGGREGATE_TIMESTAMP }
(<source_increment_by>, <target_increment_by>, <min_value>, <max_value>)

Syntax Elements

source_series_table Defines the name of one or two equidistant series tables with the SERIES TABLE syntax.

<source_series_table> ::= <identifier>
<generate_series_table> ::= <identifier>

value_range Defines the range of values.

/value_range> ::= <min_value> [, <max_value>]
/min_value> ::= <real_const> | <datetime_const>
/max_value> ::= <real_const> | <datetime_const>

When using the SERIES TABLE syntax, the values of the source increment and/or generated increment are retrieved from the series definitions of the respective table references. The <min_value> is the greater <min_value> of the two specified tables, and the <max_value> is the lesser <max_value> of the two specified tables. The <min_value> and <max_value> can still be manually specified if a SERIES TABLE reference is supplied. Manually specified values override those in the series table definition. The generic form of the procedure that takes at least one SERIES TABLE reference does not have to use the name of the series data type in the procedure name.

source_increment_by Sets the source increment value.

/source_increment_by> ::= <real_const> | <datetime_const>

generate_series_table Generates the series table with the SERIES TABLE syntax.

<generate_series_table> ::= <identifier>

target_increment_by Sets the target increment value.

/target_increment_by> ::= <real_const> | <datetime_const>

constant_literal The <constant_literal> parameter is defined as a <real_const> or a <datetime_const>.

/constant_literal> ::= <real_const> | <datetime_const>

It is either an integer constant, an interval constant, or a defined number of elements between a minimum and maximum. If the period type is a DATETIME, then the number needs to be preceded by INTERVAL followed by a time unit, such as YEAR, MONTH, DAY, HOUR, MINUTE, or SECOND. Exponential notation is allowed.
Description

Generates a complete series table with rows disaggregated into partitions defined by the `<generate_increment_by>` parameter.

The resulting table can be joined with a series table to perform horizontal disaggregation.

For date/time types, the `<increment_by>` parameter should be a string in the form "INTERVAL number units" where units is either YEAR, MONTH, DAY, HOUR, MINUTE, or SECOND and the number must be an integer, unless the units type is SECOND and the associated data type is TIMESTAMP.

The `<min_value>` parameter for the numeric versions of this function does not need to be aligned to numeric zero with respect to the `<increment_by>` value. You can specify a `<source_increment>` of 2 and a `<min_value>` of 1, which results in a source series of 1, 3, 5, and so on. This behavior differs from the SERIES TABLE descriptor, which requires that the values of `<min_value>` and `<max_value>` be aligned to zero with respect to the `<increment_by>` value.

This function returns a table with the following columns:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCE_PERIOD_START</td>
<td>PERIOD_TYPE</td>
<td>Specifies the start of the source period that generated this row.</td>
</tr>
<tr>
<td>SOURCE_PERIOD_END</td>
<td>PERIOD_TYPE</td>
<td>Specifies the end of the source period that generated this row.</td>
</tr>
<tr>
<td>GENERATED_PERIOD_START</td>
<td>PERIOD_TYPE</td>
<td>Displays the start of the period represented by this row, including the period_start. This value is a closed interval at the start.</td>
</tr>
<tr>
<td>GENERATED_PERIOD_END</td>
<td>PERIOD_TYPE</td>
<td>Displays the end of the period represented by this row as an open interval. The period represented by this row consists of all times that are greater than or equal to the start and less than the end.</td>
</tr>
<tr>
<td>ELEMENT_NUMBER_IN_SOURCE_PERIOD</td>
<td>BIGINT</td>
<td>Specifies the element number of this period within its source interval.</td>
</tr>
<tr>
<td>ELEMENT_NUMBER_IN_GENERATED_SERIES</td>
<td>BIGINT</td>
<td>Specifies the element number within the whole result set.</td>
</tr>
<tr>
<td>FRACTION_OF_SOURCE_PERIOD</td>
<td>DOUBLE</td>
<td>Specifies the fraction of the length of the source period that this generated period covers.</td>
</tr>
<tr>
<td>FRACTION_OF_MIN_MAX_RANGE</td>
<td>DOUBLE</td>
<td>Specifies the fraction of the length of all generated periods that this generated period covers.</td>
</tr>
</tbody>
</table>
Examples

The fictitious example below illustrates how to use disaggregation when inserting data into a series table:

```sql
CREATE COLUMN TABLE sourceseries(id INT, ts TIMESTAMP, val DECIMAL(8,2)) SERIES(SERIES KEY(id) EQUIDISTANT INCREMENT BY INTERVAL 1 YEAR MINVALUE '1999-01-01' MAXVALUE '2003-01-01' PERIOD FOR SERIES (ts));
CREATE COLUMN TABLE targetseries(id INT, ts TIMESTAMP, val DECIMAL(8,2)) SERIES(SERIES KEY(id) EQUIDISTANT INCREMENT BY INTERVAL 3 MONTH MINVALUE '1999-01-01' MAXVALUE '2001-01-01' PERIOD FOR SERIES (ts));
INSERT INTO targetseries(id, ts, val)
SELECT id, GENERATED_PERIOD_START AS ts, val * FRACTION_OF_SOURCE_PERIOD AS val
FROM SERIES_DISAGGREGATE(
    SERIES TABLE sourceseries, SERIES TABLE targetseries)
SD JOIN sourceseries S ON source_period_start = ts ORDER BY id, ts;
```

The example below generates a series of dates ranging from 1999-01-01 to 2001-01-04:

```sql
SELECT * FROM SERIES_DISAGGREGATE_DATE
('INTERVAL 1 year', 'INTERVAL 3 MONTH', '1999-01-01', '2001-01-04' );
```

The following result is returned by the example above:

<table>
<thead>
<tr>
<th>SOURCE_PERIOD_START</th>
<th>SOURCE_PERIOD_END</th>
<th>GENERATED_PERIOD_START</th>
<th>GENERATED_PERIOD_END</th>
<th>ELEMENT_NUMBER_IN_SOURCE_PERIOD</th>
<th>ELEMENT_NUMBER_IN_GENERATED_SERIES</th>
<th>FRACTION_OF_SOURCE_PERIOD</th>
<th>FRACTION_OF_MIN_MAX_RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 1, 1999</td>
<td>Jan 1, 2000</td>
<td>Jan 1, 1999</td>
<td>Apr 1, 1999</td>
<td>1</td>
<td>1</td>
<td>0.246575342</td>
<td>4657534</td>
</tr>
<tr>
<td>Jan 1, 1999</td>
<td>Jan 1, 2000</td>
<td>Apr 1, 1999</td>
<td>Jul 1, 1999</td>
<td>1</td>
<td>2</td>
<td>0.249315068</td>
<td>4931507</td>
</tr>
<tr>
<td>Jan 1, 1999</td>
<td>Jan 1, 2000</td>
<td>Jul 1, 1999</td>
<td>Oct 1, 1999</td>
<td>1</td>
<td>3</td>
<td>0.252054794</td>
<td>52054796</td>
</tr>
<tr>
<td>Jan 1, 1999</td>
<td>Jan 1, 2000</td>
<td>Oct 1, 1999</td>
<td>Jan 1, 2000</td>
<td>1</td>
<td>4</td>
<td>0.252054794</td>
<td>52054796</td>
</tr>
<tr>
<td>Jan 1, 2000</td>
<td>Jan 1, 2001</td>
<td>Jan 1, 2000</td>
<td>Apr 1, 2000</td>
<td>2</td>
<td>5</td>
<td>0.248633879</td>
<td>78142076</td>
</tr>
<tr>
<td>Jan 1, 2000</td>
<td>Jan 1, 2001</td>
<td>Apr 1, 2000</td>
<td>Jul 1, 2000</td>
<td>2</td>
<td>6</td>
<td>0.248633879</td>
<td>78142076</td>
</tr>
<tr>
<td>Jan 1, 2000</td>
<td>Jan 1, 2001</td>
<td>Jul 1, 2000</td>
<td>Oct 1, 2000</td>
<td>2</td>
<td>7</td>
<td>0.251366120</td>
<td>21857924</td>
</tr>
</tbody>
</table>
The example below returns the same result set as the previous example:

```
CREATE COLUMN TABLE sourceseries(id  INT, ts TIMESTAMP, val DECIMAL(8,2))     SERIES(SERIES KEY(id ) EQUIDISTANT INCREMENT BY INTERVAL 1 YEAR
               MINVALUE '1999-01-01'
               MAXVALUE '2003-01-01'
               PERIOD FOR SERIES (ts));
CREATE COLUMN TABLE targetseries(id  INT, ts TIMESTAMP, val DECIMAL(8,2))     SERIES(SERIES KEY(id ) EQUIDISTANT INCREMENT BY INTERVAL 3 MONTH
               MINVALUE '1999-01-01'
               MAXVALUE '2001-01-01'
               PERIOD FOR SERIES (ts));
SELECT * from SERIES_DISAGGREGATE(     SERIES TABLE sourceseries, SERIES TABLE targetseries);
```

The most restrictive values of the min/max in the series table references are used.

The example below illustrates how to create a series table and perform a horizontal disaggregation:

```
CREATE COLUMN TABLE testdata AS (      SELECT g.*, CAST(RAND() * 10 AS DECIMAL(5, 2)) AS val
                                      FROM SERIES_GENERATE_DECIMAL(1,0,10000) g);
SELECT s.ELEMENT_NUMBER, s.val,
       g.ELEMENT_NUMBER_IN_GENERATED_SERIES,
       s.val * g.FRACTION_OF_SOURCE_PERIOD AS DA
FROM testdata AS s
       LEFT JOIN SERIES_DISAGGREGATE_DECIMAL(1, 0.5, 0, 10000) g
       ON s.GENERATED_PERIOD_START = g.SOURCE_PERIOD_START;
```

The example above returns a result similar to the following:

<table>
<thead>
<tr>
<th>ELEMENT_NUMBER</th>
<th>VAL</th>
<th>ELEMENT_NUMBER_IN_GENERATED_SERIES</th>
<th>DA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.21</td>
<td>1</td>
<td>1.105</td>
</tr>
<tr>
<td>1</td>
<td>2.21</td>
<td>2</td>
<td>1.105</td>
</tr>
<tr>
<td>2</td>
<td>5.52</td>
<td>3</td>
<td>2.76</td>
</tr>
<tr>
<td>2</td>
<td>5.52</td>
<td>4</td>
<td>2.76</td>
</tr>
<tr>
<td>3</td>
<td>2.35</td>
<td>5</td>
<td>1.175</td>
</tr>
<tr>
<td>3</td>
<td>2.35</td>
<td>6</td>
<td>1.175</td>
</tr>
</tbody>
</table>
4.9.1.172 SERIES_ELEMENT_TO_PERIOD Function (Series Data)

Returns the series period value associated with the specified one-based series element number.

Syntax

SERIES_ELEMENT_TO_PERIOD( <element_number>, { <increment_by>, <min_value>, <max_value> | SERIES TABLE <series_table> } )

Syntax Elements

- **element_number**: Specifies the element number.

  ```
  <element_number> ::= INTEGER
  ```

- **increment_by**: Defined as either an integer constant, an interval constant, or a defined number of elements between a minimum and a maximum.

  ```
  <increment_by> ::= <real_const> | <interval_const>
  ```

  If the period type is a DATETIME, then the number needs to be preceded by INTERVAL followed by a time unit, such as YEAR, MONTH, DAY, HOUR, MINUTE, or SECOND. Exponential notation is allowed.

- **min_value**: Specifies the minimum value. It can either be a numeric or a date/time value.

  ```
  <min_value> ::= <real_const> | <datetime_const>
  ```

- **max_value**: Specifies the maximum value. It can either be a numeric or a date/time value.

  ```
  <max_value> ::= <real_const> | <datetime_const>
  ```

- **series_table**: If a series table reference is used, then the <increment_by>, <min_value>, and <max_value> values from the SERIES TABLE definition are used as the parameters for this function.

  ```
  <series_table> ::= <identifier>
  ```
Returns the series period value associated with the provided one-based series element number, where rounded_period = SERIES_ROUND (period, interval, round_mode) and element = 1 + ( rounded_period - min_value ) / interval.

Examples

The example below picks the fifth element from a series from 0 to 10 in increments of 2 and returns 8:

```
SELECT SERIES_ELEMENT_TO_PERIOD(5, 2, 0, 10) "val" FROM DUMMY;
```

The example below picks the sixth element from a series from 1 to 10 in increments of 1.25 and returns 7.25:

```
SELECT SERIES_ELEMENT_TO_PERIOD(6, 1.25, 1, 10) "val" FROM DUMMY;
```

The example below picks the seventh element from a date series from 2014-01-01 to 2014-12-31 in increments of 1 day and returns Jan 7, 2014:

```
SELECT SERIES_ELEMENT_TO_PERIOD(7, 'INTERVAL 1 DAY', '2014-01-01', '2014-12-31')      "val" FROM DUMMY;
```

The previous example can be written as follows to refer to an equidistant series table, and returns 7:

```
CREATE COLUMN TABLE ExampleSeriesTable(id INTEGER, ts TIMESTAMP)     SERIES(SERIES KEY(id) EQUIDISTANT INCREMENT BY INTERVAL 1 DAY MINVALUE '2014-01-01' MAXVALUE '2014-12-31' PERIOD FOR SERIES(ts));
SELECT SERIES_PERIOD_TO_ELEMENT(      '2014-01-07', SERIES TABLE ExampleSeriesTable) "val" FROM DUMMY;
```

The example below picks the seventh element from a date series from 2014-01-01 to 2014-12-31 in increments of 1 month and returns Jul 1, 2014:

```
SELECT SERIES_ELEMENT_TO_PERIOD(7, 'INTERVAL 1 MONTH',     '2014-01-01', '2014-12-31') "val" FROM DUMMY;
```

The example below picks the 500,000th element from a time series from 2014-01-01 to 2014-12-31 in increments of 1.5 seconds and returns Jan 9, 2014 4:19:58.5 PM:

```
SELECT SERIES_ELEMENT_TO_PERIOD(500000, 'INTERVAL 1.5 SECOND',     '2014-01-01 00:00:00.000', '2014-12-31') "val" FROM DUMMY;
```
4.9.1.173 SERIES_FILTER Function (Window)

Performs a filtered calculation on a series of data.

Syntax

```sql
SERIES_FILTER(<filter_parameter> => <expression> [, <filter_parameter> => <expression> ... ]) OVER ( [ <series_definition> | <series_reference> ] [ window_partition_by_clause ] [ ORDER BY <window_order_by_expression> ] [ window_frame_clause ] )
```

Syntax Elements

- `filter_parameter`
  - `<filter_parameter> ::= VALUE | METHOD_NAME | ALPHA | BETA`
  - VALUE is a required parameter that specifies the column to which the filter is applied. The column must represent numeric data. NULL values are not permitted.
  - METHOD_NAME is a required parameter that specifies the name of the filter method. Specify SINGLESMOOTH for single exponential smoothing or DOUBLESMOOTH for double exponential smoothing.
  - ALPHA is an optional smoothing decimal parameter for the level of the series that ranges between 0 and 1. The default value is 0.1.
  - BETA is a smoothing decimal parameter for the trend of the series that ranges between 0 and 1. This parameter is only required for double exponential smoothing.

- `window_partition_by_clause`
  - Specifies one or more partitions to be created using the supplied expressions.

- `window_order_by_expression`
  - Determines the order of the input. Only one expression must be supplied in the `window_order_by_expression` for this function.
This expression must not contain any null values or duplicates. See the description of the `<window_order_by_clause>` clause in the Window Functions topic for other syntax information for this clause.

**window_frame_clause**

Physically limits the rows within a partition by specifying a number of rows preceding or following the current row.

```
<window_frame_clause> ::= <window_frame_unit> <window_frame_extent>
<window_frame_unit> ::= ROWS
<window_frame_extent> ::= <window_frame_start> | <window_frame_between>
<window_frame_start> ::= UNBOUNDED PRECEDING | <window_frame_preceding> | CURRENT ROW
<window_frame_preceding> ::= <unsigned_integer> PRECEDING
>window_frame_between ::= BETWEEN <lower_window_frame_bound> AND <upper_window_frame_bound>
<lower_window_frame_bound> ::= <window_frame_bound>
<upper_window_frame_bound> ::= <window_frame_bound>
>window_frame_bound ::= <window_frame_start> | UNBOUNDED FOLLOWING |
>window_frame_following ::= <unsigned_integer> FOLLOWING
```

ROWS requires a ORDER BY clause to be specified.

UNBOUND PRECEDING specifies that the window starts at the first row of the partition. It can only be specified as `<window_frame_start>`. CURRENT ROW specifies that the window starts or ends at the current row when used with ROWS.

PRECEDING indicates that the number of rows or values to precede the current row. `<unsigned_integer>` should be 0 or a positive integer literal.

`<lower_window_frame_bound>` and `<upper_window_frame_bound>` specify the lower (starting) bound and the upper (ending) bound of the window frame. The upper bound cannot be smaller than the lower bound.

UNBOUNDED FOLLOWING specifies that the window ends at the last row of the partition. UNBOUNDED FOLLOWING can only be specified as a window frame end.

FOLLOWING indicates the number of rows or values to follow the current row. When specified as the window frame start, the window frame end should be `<unsigned_integer>` FOLLOWING. The `<unsigned_integer>` should be 0 or a positive integer literal.

**Description**

The window functions allow you to divide the result sets of a query, or a logical partition of a query, into groups of rows called window partitions.

A window partition is specified by one or more expressions in the OVER clause.

Window functions other than ROW_NUMBER, PERCENTILE_DISC, PERCENTILE_CONT and the window aggregate functions must have an ORDER BY clause in the OVER clause.

Result sets are first partitioned as specified by the PARTITION BY clause, and then sorted by ORDER BY clause specification within the window partition. Then, window functions are applied to each row within window partition boundaries.
The ORDER BY clause in the OVER clause only evaluates the window function so that the order of resulting rows is non-deterministic if not specified by ORDER BY for SELECT.

The default window frame of the window function depends on whether a window ORDER BY clause is specified. If a window ORDER BY clause is specified, then the default window frame becomes 'between UNBOUNDED PRECEDING and CURRENT ROW': The window function computes on rows preceding or peer with the current row. As a result, the function returns cumulative values.

If a window ORDER BY clause is not specified, then the default window frame becomes 'between UNBOUNDED PRECEDING and UNBOUNDED FOLLOWING': The window function computes on the whole window partition. As a result, the function returns the same value within a window partition regardless of current row.

Examples

The following example illustrates single and double exponential smoothing with forecasting.

```
DROP TABLE weather;
CREATE COLUMN TABLE weather (ts DATE, temperature FLOAT);
INSERT INTO weather VALUES('2014-01-01', 0);
INSERT INTO weather VALUES('2014-01-02', 3);
INSERT INTO weather VALUES('2014-01-03', 4.5);
INSERT INTO weather VALUES('2014-01-04', 6);
INSERT INTO weather VALUES('2014-01-05', 6.3);
INSERT INTO weather VALUES('2014-01-06', 6.9);
INSERT INTO weather VALUES('2014-01-07', NULL);
INSERT INTO weather VALUES('2014-01-08', NULL);
SELECT ts, temperature,
    SERIES_FILTER(VALUE => temperature, METHOD_NAME => 'SINGLESMOOTH', ALPHA => 0.2) OVER (ORDER BY ts) AS SES,
    SERIES_FILTER(VALUE => temperature, METHOD_NAME => 'DOUBLESMOOTH', ALPHA => 0.2, BETA => 0.3) OVER (ORDER BY ts) AS DES
FROM weather;
```

<table>
<thead>
<tr>
<th>TS</th>
<th>TEMPERATURE</th>
<th>SES</th>
<th>DES</th>
</tr>
</thead>
<tbody>
<tr>
<td>01.01.2014</td>
<td>0.0</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>02.01.2014</td>
<td>3.0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>03.01.2014</td>
<td>4.5</td>
<td>0.60000000000000001</td>
<td>6</td>
</tr>
<tr>
<td>04.01.2014</td>
<td>6.0</td>
<td>1.38000000000000001</td>
<td>8.61</td>
</tr>
<tr>
<td>05.01.2014</td>
<td>6.3</td>
<td>2.30400000000000003</td>
<td>10.8414</td>
</tr>
<tr>
<td>06.01.2014</td>
<td>6.9</td>
<td>3.1032</td>
<td>12.414036</td>
</tr>
<tr>
<td>07.01.2014</td>
<td>NULL</td>
<td>3.86256</td>
<td>13.46130264</td>
</tr>
<tr>
<td>08.01.2014</td>
<td>NULL</td>
<td>3.86256</td>
<td>15.61137648</td>
</tr>
</tbody>
</table>

The following example uses a window frame ROWS specification:

```
CREATE ROW TABLE T (class CHAR(10), val INT, offset INT);
INSERT INTO T VALUES('A', 1, 1);
```
```
INSERT INTO T VALUES('A', 3, 3);
INSERT INTO T VALUES('A', 5, null);
INSERT INTO T VALUES('A', 5, 2);
INSERT INTO T VALUES('A', 10, 0);
INSERT INTO T VALUES('B', 1, 3);
INSERT INTO T VALUES('B', 1, 1);
INSERT INTO T VALUES('B', 7, 1);
SELECT class,
    val,
    SUM(val) OVER(PARTITION BY class ORDER BY val) AS s0,
    SUM(val) OVER(PARTITION BY class ORDER BY val ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS s1,
    SUM(val) OVER (PARTITION BY class ORDER BY val ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS s2,
    SUM(val) OVER (PARTITION BY class ORDER BY val ROWS BETWEEN UNBOUNDED PRECEDING AND 1 PRECEDING) AS s3,
    SUM(val) OVER (PARTITION BY class ORDER BY val ROWS BETWEEN UNBOUNDED PRECEDING AND 1 FOLLOWING) AS s4,
    SUM(val) OVER (PARTITION BY class ORDER BY val ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS s5,
    SUM(val) OVER (PARTITION BY class ORDER BY val ROWS BETWEEN CURRENT ROW AND CURRENT ROW) AS s6,
    SUM(val) OVER (PARTITION BY class ORDER BY val ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING) AS s7
FROM T;
```

<table>
<thead>
<tr>
<th>CLA</th>
<th>SS</th>
<th>VAL</th>
<th>S0</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
<th>S9</th>
<th>S10</th>
<th>S11</th>
<th>S12</th>
<th>S13</th>
<th>S14</th>
<th>S15</th>
<th>S16</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>24</td>
<td>1</td>
<td>null</td>
<td>4</td>
<td>24</td>
<td>1</td>
<td>4</td>
<td>24</td>
<td>1</td>
<td>null</td>
<td>4</td>
<td>23</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>4</td>
<td>24</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>23</td>
<td>3</td>
<td>8</td>
<td>24</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>20</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>14</td>
<td>24</td>
<td>9</td>
<td>4</td>
<td>14</td>
<td>20</td>
<td>5</td>
<td>10</td>
<td>23</td>
<td>8</td>
<td>4</td>
<td>13</td>
<td>15</td>
<td>15</td>
<td>9</td>
<td>5</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>14</td>
<td>24</td>
<td>14</td>
<td>9</td>
<td>24</td>
<td>15</td>
<td>5</td>
<td>15</td>
<td>20</td>
<td>10</td>
<td>8</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>14</td>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>14</td>
<td>24</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>10</td>
<td>15</td>
<td>null</td>
<td>null</td>
<td>24</td>
<td>10</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>null</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>null</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>9</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>9</td>
<td>7</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>null</td>
<td>null</td>
<td>9</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Related Information

Expressions [page 56]
Window Functions and the Window Specification [page 424]
4.9.1.174 SERIES_GENERATE Function (Series Data)

Generates a complete series table based on the specified series definition.

Syntax

```
SERIES_GENERATE( SERIES TABLE <table_name> [, <min_value> [, <max_value> ]] )
| SERIES_GENERATE_<generation_spec>
```

Syntax Elements

generation_spec

A version of the function that specifies the data type for the generated parameters.

```
<generation_spec> ::= <data_type> <generate_parameters>
```

data_type

The type of data to use for the series.

```
<data_type> ::= TINYINT
| SMALLINT
| INTEGER
| BIGINT
| SMALLDECIMAL
| DECIMAL
| TIME
| DATE
| SECONDDATE
| TIMESTAMP
```

generate_parameters

Specifies the increment value used to generate parameters.

```
<generate_parameters> ::= { <increment_by>, <min_value>, <max_value> }
<increment_by> ::= { <numeric_literal> | INTERVAL <date_literal> }
```

<increment_by> defines the period size. If the period type is a DATETIME, then the number needs to be preceded by INTERVAL followed by a time unit, such as YEAR, MONTH, DAY, HOUR, MINUTE, or SECOND.

table_name

Specifies the name of the series table containing the series definition.

```
<table_name> ::= <identifier>
```

min_value
Sets the minimum value of the generated series. This overrides the value defined in the series table, if one is specified.

\[ \text{min\_value} ::= \text{numeric\_literal} | \text{date\_literal} \]

**max\_value**

Sets the maximum value of the generated series. This overrides the value defined in the series table, if one is specified.

\[ \text{max\_value} ::= \text{numeric\_literal} | \text{date\_literal} \]

**Description**

The range is defined by the \text{min\_value} and \text{max\_value} parameters, and the period size is defined by the \text{increment\_by} parameter. The \text{min\_value} and \text{max\_value} specifies the increment value used to generate values must be aligned with the \text{increment\_by} value. The generated intervals have closed-open semantics, so the GENERATED\_PERIOD\_END does not belong to the interval.

Alternatively, the name of an existing series table can be defined with the SERIES TABLE syntax. In this case, the values of the increment, \text{min\_value}, and \text{max\_value} are retrieved from the series definition associated with that table. The \text{min\_value} and \text{max\_value} can still be manually specified if a SERIES TABLE reference is supplied; the manually specified values override those in the series table definition. The generic form of the procedure that takes at least one SERIES TABLE reference does not have to use the name of the series data type in the procedure name.

The \text{min\_value} parameter for the numeric versions of this function does not need to be aligned to numeric zero with respect to the \text{increment\_by} value. You can specify a source increment of 2 and a \text{min\_value} of 1, which results in a source series of 1, 3, 5, and so on. This behavior differs from the SERIES TABLE descriptor, which requires that the values of \text{min\_value} and \text{max\_value} be aligned to zero with respect to the \text{increment\_by} value.

This function returns a table with the following columns:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERATED_PERIOD_START</td>
<td>PERIOD_TYPE</td>
<td>Specifies the start of the period represented by this row, including the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\text{period_start} parameter. This is a closed interval at the start.</td>
</tr>
<tr>
<td>GENERATED_PERIOD_END</td>
<td>PERIOD_TYPE</td>
<td>Specifies the end of the period represented by this row as an open interval.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The period represented by this row consists of all times that are greater</td>
</tr>
<tr>
<td></td>
<td></td>
<td>than or equal to the start, and less than the end.</td>
</tr>
<tr>
<td>ELEMENT_NUMBER</td>
<td>BIGINT</td>
<td>Specifies the element number of this period within the generated series.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This value is equivalent to the result set’s row number.</td>
</tr>
</tbody>
</table>
**FRACTION_OF_MIN_MAX_RANGE**  DOUBLE

Specifies the ratio of the length of this period to the length of all periods generated by this function. The sum of FRACTION_OF_MIN_MAX_RANGE is close to 1.0. For fixed size intervals, such as 1 DAY, the value is computed as 1/numPeriods. For non-fixed size intervals, such as MONTH and YEAR, the value is computed individually for each row, which is calculated by the time span of the interval divided by the time span of the entire result.

**Examples**

The example below illustrates how to generate a series table:

```
CREATE COLUMN TABLE MyTab (     profile_id INT,     ts TIMESTAMP,     consumption DECIMAL(4,3))     SERIES(         SERIES KEY(profile_id)         PERIOD FOR SERIES(ts)         EQUIDISTANT INCREMENT BY INTERVAL 1 HOUR MISSING ELEMENTS ALLOWED         MINVALUE '2010-01-01'         MAXVALUE '2015-01-01'); SELECT * FROM SERIES_GENERATE_TIMESTAMP(SERIES TABLE MyTab);
```

The example below generates a series of decimals ranging from 0 to 10 that increments by 2.5:

```
SELECT * FROM SERIES_GENERATE_DECIMAL(2.5, 0, 10);
```

The following result is returned by the example above:

<table>
<thead>
<tr>
<th>GENERATED_PERIOD_START</th>
<th>GENERATED_PERIOD_END</th>
<th>ELEMENT_NUMBER</th>
<th>FRACTION_OF_MIN_MAX_RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.5</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>2.5</td>
<td>5</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>5</td>
<td>7.5</td>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>7.5</td>
<td>10</td>
<td>4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

The example below generates a series of integers ranging from 1 to 5 that increments by 2:

```
SELECT * FROM SERIES_GENERATE_INTEGER(2, 1, 5);
```
The following result is returned by the example above:

<table>
<thead>
<tr>
<th>GENERATED_PERIOD_START</th>
<th>GENERATED_PERIOD_END</th>
<th>ELEMENT_NUMBER</th>
<th>FRAC-TION_OF_MIN_MAX_RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>0.5</td>
</tr>
</tbody>
</table>

The example below illustrates an equivalent way of writing the previous example using the SERIES TABLE syntax and a series table named MySeries:

```sql
CREATE COLUMN TABLE MySeries(id INTEGER, pos INTEGER) SERIES (SERIES KEY(id) EQUIDISTANT INCREMENT BY 1 MINVALUE 1 MAXVALUE 5 PERIOD FOR SERIES(pos));
SELECT * FROM SERIES_GENERATE(SERIES_TABLE MySeries);
```

The example below generates a series of timestamps ranging from 1999-01-01 to 1999-01-02 that increments by 30 second intervals:

```sql
SELECT * FROM SERIES_GENERATE_TIMESTAMP('INTERVAL 30 SECOND', '1999-01-01', '1999-01-02');
```

The following result is returned by the example above:

<table>
<thead>
<tr>
<th>GENERATED_PERIOD_START</th>
<th>GENERATED_PERIOD_END</th>
<th>ELEMENT_NUMBER</th>
<th>FRAC-TION_OF_MIN_MAX_RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 1, 1999 12:00:00.0 AM</td>
<td>Jan 1, 1999 12:00:30.0 AM</td>
<td>1</td>
<td>0.003472222222222224</td>
</tr>
<tr>
<td>Jan 1, 1999 12:00:30.0 AM</td>
<td>Jan 1, 1999 12:01:00.0 AM</td>
<td>2</td>
<td>0.003472222222222224</td>
</tr>
<tr>
<td>Jan 1, 1999 12:01:00.0 AM</td>
<td>Jan 1, 1999 12:01:30.0 AM</td>
<td>3</td>
<td>0.003472222222222224</td>
</tr>
<tr>
<td>Jan 1, 1999 12:01:30.0 AM</td>
<td>Jan 1, 1999 12:02:00.0 AM</td>
<td>4</td>
<td>0.003472222222222224</td>
</tr>
</tbody>
</table>

The example below illustrates how to implement a series of timestamps ranging from 1999-01-01 to 1999-01-02 that increments by 30 second intervals using a series table named testseries:

```sql
CREATE COLUMN TABLE testseries(id INT, ts TIMESTAMP, val DOUBLE) SERIES (SERIES KEY(id) EQUIDISTANT INCREMENT BY INTERVAL 30 SECOND MINVALUE '1999-01-01' MAXVALUE '1999-01-02' PERIOD FOR SERIES (ts));
SELECT * FROM SERIES_GENERATE(SERIES_TABLE testseries);
```

The example below illustrates how to implement a series of timestamps ranging from 1999-01-01 to 1999-01-02 that increments by 30 second intervals using a series table named testseries:
The example below illustrates how to generate a date series table with closed-closed semantics where the period end belongs to the interval:

```
SELECT GENERATED_PERIOD_START AS from_date,
      ADD_DAYS(GENERATED_PERIOD_END, -1) AS to_date
FROM SERIES_GENERATE_DATE('INTERVAL 1 MONTH', '2010-01-01', '2011-01-01')
```

### 4.9.1.175 SERIES_PERIOD_TO_ELEMENT Function (Series Data)

Returns the one-based series element number that the given period value is associated with.

#### Syntax

```
SERIES_PERIOD_TO_ELEMENT(    <value>, {<increment_by>, <min_value>, <max_value> [, <rounding_mode>]   | SERIES TABLE <series_table> [, <rounding_mode>]}   )
```

#### Syntax Elements

- **value** Specifies that the `<value>` parameter can either be a numeric value or a date/time type.
  ```
  <value> ::= INTEGER | DOUBLE | TIMESTAMP
  ```
- **increment_by** Sets the increment value.
  ```
  <increment_by> ::= <identifier>
  ```
  If the `<value>` parameter is a date/time type, then the `<increment_by>` parameter should be a string in the form "INTERVAL number units" where units is either YEAR, MONTH, DAY, HOUR, MINUTE, or SECOND and the number must be an integer, unless the units type is SECOND and the associated data type is TIMESTAMP. If the value is numeric, then the `<increment_by>` parameter should be a numeric value that defines the period of the series.
- **min_value** Sets the minimum period value.
  ```
  <min_value> ::= <integer>
  ```
  For an equidistant series, there is a mapping between periods and elements. The periods are in the space of the period columns, usually timestamps. The elements are always BIGINT. The element 1 represents the first period, associated with `<min_value>`.
- **max_value** Sets the maximum period value.
  ```
  <max_value> ::= <integer>
  ```
**rounding_mode** Specifies that if the period value is not on a series period value, then it is rounded to the nearest series period value, which returns the associated element number.

```sql
<rounding_mode> ::= ROUND_HALF_UP  | ROUND_HALF_DOWN  | ROUND_HALF_EVEN  | ROUND_UP  | ROUND_DOWN  | ROUND_CEILING  | ROUND_FLOOR
```

The supported rounding modes are:

<table>
<thead>
<tr>
<th>Rounding mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROUND_HALF_UP</td>
<td>Specifies that the value is rounded to the nearest series value. Values that fall halfway between two series values are rounded up away from zero. This is the default value.</td>
</tr>
<tr>
<td>ROUND_HALF_DOWN</td>
<td>Specifies that the value is rounded to the nearest series value. Values that fall halfway between two round values are rounded down toward zero.</td>
</tr>
<tr>
<td>ROUND_HALF_EVEN</td>
<td>Specifies that the value is rounded to the nearest series value. Values that fall halfway between two rounded values are rounded to the even series value based on element number.</td>
</tr>
<tr>
<td>ROUND_UP</td>
<td>Specifies that the value is always rounded away from zero to the larger series value.</td>
</tr>
<tr>
<td>ROUND_DOWN</td>
<td>Specifies that the value is always rounded towards zero to the smaller series value.</td>
</tr>
<tr>
<td>ROUND_CEILING</td>
<td>Specifies that the value is always rounded in a positive direction to the larger series value.</td>
</tr>
<tr>
<td>ROUND_FLOOR</td>
<td>Specifies that the value is always rounded in a negative direction to the smaller series value.</td>
</tr>
</tbody>
</table>

**series_table** If a series table reference is used, then this specifies that the values for `<increment_by>`, `<min_value>`, and `<max_value>` from the SERIES TABLE definition are used as the parameters for this function.

```sql
<series_table> ::= <identifier>
```

**Description**

Returns the one-based series element number with which the given period value is associated, where period = min_value + (element - 1) * interval.
Examples

The following example returns the result 4:

```
SELECT SERIES_PERIOD_TO_ELEMENT(5, 2, 0, 10, ROUND_HALF_UP) "element" FROM DUMMY;
```

The following example returns the result 3:

```
SELECT SERIES_PERIOD_TO_ELEMENT(5, 2, 0, 10, ROUND_HALF_DOWN) "element" FROM DUMMY;
```

The example below picks the next element rounded down from 2014-01-05 12:00:00 from a date series from 2014-01-01 to 2014-12-32 in increments of 1 day. It returns the result 5:

```
SELECT SERIES_PERIOD_TO_ELEMENT('2014-01-05 12:00:00', 'INTERVAL 1 DAY', '2014-01-01', '2014-12-31', ROUND_HALF_DOWN) "element" FROM DUMMY;
```

The previous example could be written like the following fictitious example to refer to an equidistant series table:

```
CREATE COLUMN TABLE ExampleSeriesTable(id INTEGER, ts TIMESTAMP) SERIES(SERIES KEY(id) EQUIDISTANT INCREMENT BY INTERVAL 1 DAY MINVALUE '2014-01-01' MAXVALUE '2014-12-31' PERIOD FOR SERIES(ts));
SELECT SERIES_PERIOD_TO_ELEMENT('2014-01-05 12:00:00', SERIES TABLE ExampleSeriesTable, ROUND_HALF_DOWN) "element" FROM DUMMY;
```

4.9.1.176  SERIES_ROUND Function (Series Data)

Rounds a specified value to the series value using the specified rounding settings.

Syntax

```
SERIES_ROUND( <value>, { <increment_by> | SERIES TABLE <series_table> } [, <rounding_mode> [, <alignment_expression> ] ] )
```

Syntax Elements

- **value**: Specifies that the value parameter can either be a numeric value or a date/time type.

```
<value> ::= { <real_const> | <datetime_const> }
```
increment_by If the `<value>` parameter is numeric, then the `<increment_by>` value must be a numeric value that defines the period of the series.

```plaintext
<increment_by>::= <interval_const>
```

If the `<value>` parameter is a date/time type, then the `<increment_by>` value must be a string of the form "INTERVAL number units" where units is either YEAR, MONTH, DAY, HOUR, MINUTE, or SECOND and the number must be an integer, unless the unit type is SECOND and the associated data type is TIMESTAMP.

**series_table** Specifies that the maximum and minimum values in a provided series table are used to calculate the rounded values.

```plaintext
<series_table>::= <identifier>
```

If a SERIES TABLE reference is specified, then the `<increment_by>` value is retrieved from the series descriptor of that table. This series must be equidistant.

**rounding_mode** Specifies that the natural zero that serves as the basis for the rounding depends on the data type and interval.

```plaintext
<rounding_mode>::= ROUND_HALF_UP  |  ROUND_HALF_DOWN  |  ROUND_HALF_EVEN  
                              |  ROUND_UP        |  ROUND_DOWN       
                              |  ROUND_CEILING   |  ROUND_FLOOR      
```

For numeric types, zero is the numeric zero. For date types with the interval units of DAY or smaller, zero is midnight. For MONTH and YEAR intervals, zero is midnight on the first day of the month, and midnight on the first day of the year, respectively. The following rounding modes are supported:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROUND_HALF_UP</td>
<td>Specifies the default value. The value is rounded to</td>
</tr>
<tr>
<td></td>
<td>the nearest series value. Values that fall halfway</td>
</tr>
<tr>
<td></td>
<td>between two series values are rounded up, away from</td>
</tr>
<tr>
<td></td>
<td>zero.</td>
</tr>
<tr>
<td>ROUND_HALF_DOWN</td>
<td>Specifies that the value is rounded to the nearest</td>
</tr>
<tr>
<td></td>
<td>series value. Values that fall halfway between two</td>
</tr>
<tr>
<td></td>
<td>rounded values are rounded down, toward zero.</td>
</tr>
<tr>
<td>ROUND_HALF_EVEN</td>
<td>Specifies that the value is rounded to the nearest</td>
</tr>
<tr>
<td></td>
<td>series value. Values that fall halfway between two</td>
</tr>
<tr>
<td></td>
<td>rounded values are rounded to the even series value,</td>
</tr>
<tr>
<td></td>
<td>based on element number.</td>
</tr>
<tr>
<td>ROUND_UP</td>
<td>Specifies that the value is always rounded away from</td>
</tr>
<tr>
<td></td>
<td>zero to the larger series value.</td>
</tr>
<tr>
<td>ROUND_DOWN</td>
<td>Specifies that the value is always rounded toward</td>
</tr>
<tr>
<td></td>
<td>zero to the smaller series value.</td>
</tr>
<tr>
<td>ROUND_CEILING</td>
<td>Specifies that the value is always rounded in a</td>
</tr>
<tr>
<td></td>
<td>positive direction to the larger series value.</td>
</tr>
<tr>
<td>ROUND_FLOOR</td>
<td>Specifies that the value is always rounded in a</td>
</tr>
<tr>
<td></td>
<td>negative direction to the smaller series value.</td>
</tr>
</tbody>
</table>
alignment_expression Specifies that this parameter must be convertible to the <value> data type.

<alignment_expression> ::= { <real_const> | <datetime_const> }

A <rounding_mode> must be specified when using this parameter.

Description

Rounds a specified value to the series value using the specified rounding settings.

Examples

The example below horizontally aggregates a series table by using the ROUND_DOWN mode:

```
CREATE COLUMN TABLE ExampleSeriesTable(     sid INTEGER NOT NULL,     ts TIMESTAMP NOT NULL,     val DOUBLE)     SERIES(SERIES KEY(sid) PERIOD FOR SERIES(ts)     EQUIDISTANT INCREMENT BY INTERVAL 60 SECOND);     SELECT rounded.sid, week, AVG(val) AS weekly_avg FROM(     SELECT t.sid, SERIES_ROUND(ts, 'INTERVAL 7 DAY', ROUND_DOWN) AS week, val FROM ExampleSeriesTable AS t     ) AS rounded     GROUP BY rounded.sid, week;
```

The fictitious example below illustrates another way to perform horizontal aggregation, using the fictional table DailyWeather:

```
INSERT INTO DailyWeather(id, date, avg_temp)     SELECT id, SERIES_ROUND(ts, SERIES TABLE DailyWeather) AS date, AVG(temp) FROM HourlyWeather     GROUP BY id, SERIES_ROUND(ts, SERIES TABLE DailyWeather)     ORDER BY 1, 2;
```

The example below illustrates different ways of aggregating data when moving from narrower to wider intervals:

```
CREATE COLUMN TABLE SampleSensorData (     machine_id varchar(10),     ts timestamp,     power_consumption double,     flow_rate double,     primary key(machine_id, ts))     SERIES (     SERIES KEY(machine_id) PERIOD FOR SERIES(ts)     EQUIDISTANT INCREMENT BY INTERVAL 1 MINUTE MISSING ELEMENTS ALLOWED);-- Going from 1 minute to 1 day using Standard SQL:     SELECT machine_id,     TO_DATE(YEAR(ts) || '-' || MONTH(ts) || '-' || DAYOFMONTH(ts), 'YYYY-MM-DD') AS ts,     SUM(power_consumption) AS power_consumption FROM SampleSensorData     GROUP BY machine_id, YEAR(ts), MONTH(ts), DAYOFMONTH(ts);
```
-- Going from 1 minute to 15 minutes using SERIES_ROUND:
SELECT machine_id, ts AS original_ts,
       SERIES_ROUND(ts, 'INTERVAL 15 MINUTE', ROUND_FLOOR) AS rounded_ts
FROM SampleSensorData
WHERE machine_id = 'EQ42-P01'
ORDER BY ts;

-- Going from 1 minute to 15 minutes using SERIES_ROUND (variant #1)
SELECT machine_id, ts, SUM(power_consumption) AS power_consumption
FROM (SELECT machine_id, SERIES_ROUND(ts, 'INTERVAL 15 MINUTE', ROUND_FLOOR)
       AS ts, power_consumption
       FROM SampleSensorData
    )
GROUP BY machine_id, ts
ORDER BY machine_id, ts;

-- Going from 1 minute to 15 minutes using SERIES_ROUND (variant #2)
SELECT machine_id, SERIES_ROUND(ts, 'INTERVAL 15 MINUTE', ROUND_FLOOR)
       AS ts, sum(power_consumption) AS power_consumption
FROM SampleSensorData
GROUP BY machine_id, SERIES_ROUND(ts, 'INTERVAL 15 MINUTE', ROUND_FLOOR);

The example below shows how to round up 4.5 to a series incremented by 3. It returns the result 6:

SELECT SERIES_ROUND(4.5, 3, ROUND_HALF_UP) "result" FROM DUMMY;

The example below shows how to round down 4.5 to a series incremented by 3. The example below shows returns the result 3:

SELECT SERIES_ROUND(4.5, 3, ROUND_HALF_DOWN) "result" FROM DUMMY;

The previous example could be written as follows in the fictitious example below to refer to an equidistant series table:

CREATE COLUMN TABLE ExampleSeriesTable(id INTEGER, pos INTEGER) SERIES(SERIES KEY(id) EQUIDISTANT INCREMENT BY 3 PERIOD FOR SERIES(pos));
SELECT SERIES_ROUND(4.5, SERIES TABLE ExampleSeriesTable, ROUND_HALF_DOWN) "result" FROM DUMMY;

The example below shows how to round down a date to the beginning of the year. It returns the result Jan 1, 2013:

SELECT SERIES_ROUND('2013-05-24', 'INTERVAL 1 YEAR', ROUND_DOWN) "result" FROM DUMMY;

The example below shows how to round up a time to the next 10 minute interval. It returns the result 4:30:00 AM:

SELECT SERIES_ROUND('04:25:01', 'INTERVAL 10 MINUTE') "result" FROM DUMMY;
4.9.1.177 SESSION_CONTEXT Function (Miscellaneous)

Returns the value of the specified session variable assigned to the current user.

Syntax

```
SESSION_CONTEXT(<session_variable>)
```

Description

The `<session_variable>` can either be predefined or user-defined. Predefined session variables that can be set by the client are APPLICATION, APPLICATIONUSER, and TRACEPROFILE.

Session variables can be defined or modified using a SET [SESSION] `<variable_name>` = `<value>` statement, and unset using an UNSET [SESSION] `<variable_name>` statement.

SESSION_CONTEXT returns an NVARCHAR with a maximum length of 512 characters.

Example

The following query returns the value HDBStudio, or a similar value depending on your settings:

```
SELECT SESSION_CONTEXT('APPLICATION') "session context" FROM DUMMY;
```

Related Information

- Session Management Statements [page 1205]
- SESSION_USER Function (Miscellaneous) [page 342]
- M_SESSION_CONTEXT System View [page 2133]
- SET [SESSION] Statement (Session Management) [page 1171]
- UNSET [SESSION] Statement (Session Management) [page 1183]
4.9.1.178 SESSION_USER Function (Miscellaneous)

Returns the user name of the current session.

Syntax

SESSION_USER

Description

SESSION_USER and CURRENT_USER are similar functions, but there are differences. SESSION_USER reflects how the session connected to the server, while CURRENT_USER may be affected by the SQL SECURITY clause and reflects how permissions are applied.

Example

The following query returns the user name of the current session:

```sql
SELECT SESSION_USER "session user" FROM DUMMY;
```

Consider the following fictitious definer-mode procedure that is declared by USER_A:

```sql
CREATE PROCEDURE USER_A.PROC1 LANGUAGE SQLSCRIPT SQL SECURITY DEFINER AS
BEGIN
  SELECT SESSION_USER "session user", CURRENT_USER "current user" FROM DUMMY;
END;
```

The following fictitious query returns USER_B when USER_B executes USER_A.PROC:

```sql
CALL USER_A.PROC1;
```

Consider the following fictitious invoker-mode procedure that is declared by USER_A:

```sql
CREATE PROCEDURE USER_A.PROC2 LANGUAGE SQLSCRIPT SQL SECURITY INVOKER AS
BEGIN
  SELECT SESSION_USER "session user", CURRENT_USER "current user" FROM DUMMY;
END;
```

The following fictitious query returns USER_B when USER_B executes USER_A.PROC:

```sql
CALL USER_A.PROC2;
```
4.9.1.179 SIGN Function (Numeric)

Returns the sign (positive or negative) of the specified numeric argument.

Syntax

```
SIGN(<number>)
```

Description

Returns 1 if `<number>` is a positive value, -1 if `<number>` is a negative value, 0 if `<number>` is equal to zero, and NULL if `<number>` is equal to NULL.

Example

The following example returns the value -1 for "sign":

```
SELECT SIGN (-15) "sign" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]
4.9.1.180  SIN Function (Numeric)

Returns the sine of an angle expressed in radians.

Syntax

```
SIN(<number>)
```

Description

Returns the sine of `<number>`, where `<number>` is an angle expressed in radians.

Example

The following example returns the value 1.0 for "sine":

```
SELECT SIN (3.141592653589793/2) "sine" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.181  SINH Function (Numeric)

Returns the hyperbolic sine of an angle expressed in radians.

Syntax

```
SINH(<number>)
```
Description

Returns the hyperbolic sine of <number>, where the argument is an angle expressed in radians.

Example

The following example returns the value 0 for "sinh":

```
SELECT SINH (0.0) "sinh" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.182 SOUNDEX Function (String)

Converts alphabet characters into a sound code that represents their sound.

Syntax

```
SOUNDEX(<string>)
```

Syntax Elements

- **string** Specifies the string to be converted into sound code.

Description

Use the SOUNDEX function to find names that sound the same but are spelled differently.

SOUNDEX returns a code consisting of one letter and three numbers. The letter is the first letter of the word, and the numbers represent the next three consonants of the word, other than H, W, and Y, which are ignored. Vowels are also ignored, unless they are the first letter of the string. Double letters are counted as one letter. If necessary, zeroes are added at the end to ensure a four-character code.
The following characters are ignored:

- Blank spaces
- New lines
- Form feeds
- Carriage returns
- Line feeds
- Any of the following: ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ ] ^ _ ` { | } ~

**Example**

The following fictitious example returns two identical codes, S530, which represent the sound of each name:

```
SELECT SOUNDEX ('Smith'), SOUNDEX ('Smythe') FROM DUMMY;
```

This fictitious example searches MyTable.MyColumn for values that sound like Smith:

```
SELECT * FROM MyTable WHERE SOUNDEX(MyColumn) = SOUNDEX('SMITH');
```

**Related Information**

[Character String Data Types](page 36)

**4.9.1.183 SQRT Function (Numeric)**

Returns the square root of the specified argument.

**Syntax**

```
SQRT(<number>)
```

**Description**

Returns the square root of the numeric argument `<number>`.
Example

The following example returns the value 1.4142135623730951 for "sqrt":

```
SELECT SQRT (2) "sqrt" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.184 STDDEV Function (Aggregate)

Returns the standard deviation of the given expression as the square root of the VAR function. This function can also be used as a window function.

Syntax

Aggregate function:

```
STDDEV( [ ALL | DISTINCT ] <expression> )
```

Window function:

```
STDDEV( <expression> ) <window_specification>
```

Syntax Elements

- **expression**
  Specifies the input data for the function.
- **window_specification**
  Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].
Description

Result type based on input

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>TINYINT</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>BIGINT</th>
<th>DECIMAL (p, s)</th>
<th>DECIMAL</th>
<th>REAL</th>
<th>DOUBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECIMAL(9,6)</td>
<td>DECIMAL(1,6)</td>
<td>DECIMAL(1,6)</td>
<td>DECIMAL(2,6)</td>
<td>DECIMAL(p, s)</td>
<td>DECIMAL</td>
<td>REAL</td>
<td>DOUBLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example

The following statements create a table with data for example purposes:

```sql
DROP TABLE "MyProducts";
CREATE TABLE "MyProducts"
("Product_ID" VARCHAR(10),
"Product_Name" VARCHAR(100),
"Category" VARCHAR(100),
"Quantity" INTEGER,
"Price" DECIMAL(10,2),
PRIMARY KEY ("Product_ID")
);

INSERT INTO "MyProducts" VALUES('P1','Shirts', 'Clothes', 32, 20.99);
INSERT INTO "MyProducts" VALUES('P2','Jackets', 'Clothes', 16, 99.49);
INSERT INTO "MyProducts" VALUES('P3','Trousers', 'Clothes', 30, 32.99);
INSERT INTO "MyProducts" VALUES('P4','Coats', 'Clothes', 5, 129.99);
INSERT INTO "MyProducts" VALUES('P5','Purse', 'Accessories', 3, 89.49);
```

The following example returns 46.13, the standard deviation of the values in the Price column:

```sql
SELECT STDDEV("Price") FROM "MyProducts";
```

Related Information

Window Functions and the Window Specification [page 424]
Window Aggregate Functions [page 428]
STDDEV_POP Function (Aggregate) [page 349]
STDDEV_SAMP Function (Aggregate) [page 350]
Aggregate Functions [page 414]
Expressions [page 56]
### 4.9.1.185 STDDEV_POP Function (Aggregate)

Returns the standard deviation of a given expression as the square root of the VAR_POP function.

#### Syntax

```
STDDEV_POP(<expression>)
```

#### Description

Returns the standard deviation of the given expression as the square root of the VAR_POP function.

#### Examples

The following example returns 0 for the standard deviation of the specified expression:

```sql
CREATE ROW TABLE RTABLE (A INT);
INSERT INTO RTABLE VALUES (1);
SELECT STDDEV_POP(A) "STDDEVPOP" FROM RTABLE;
```

The following example returns 0.5 for the standard deviation of the specified expression:

```sql
INSERT INTO RTABLE VALUES (2);
SELECT STDDEV_POP(A) "STDDEVPOP" FROM RTABLE;
```

#### Related Information

- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- STDDEV Function (Aggregate) [page 347]
- STDDEV_SAMP Function (Aggregate) [page 350]
- Aggregate Functions [page 414]
4.9.1.186 STDDEV_SAMP Function (Aggregate)

Returns the standard deviation of the given expression as the square root of VAR_SAMP function.

Syntax

```
STDDEV_SAMP(<expression>)
```

Description

Returns the standard deviation of the given `<expression>` as the square root of the VAR_SAMP function.

Examples

The following example returns `NULL` for the standard deviation of the specified expression, as the square root of the VAR_SAMP function:

```
CREATE ROW TABLE RTABLE (A INT);
INSERT INTO RTABLE VALUES (1);
SELECT STDDEV_SAMP(A) "STDDEVSAMP" FROM RTABLE;
```

The following example returns `0.707107` for the standard deviation of the specified expression, as the square root of VAR_SAMP function:

```
INSERT INTO RTABLE VALUES (2);
SELECT STDDEV_SAMP(A) "STDDEVSAMP" FROM RTABLE;
```

Related Information

- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- STDDEV Function (Aggregate) [page 347]
- STDDEV_POP Function (Aggregate) [page 349]
- Aggregate Functions [page 414]
4.9.1.187 STRING_AGG Function (Aggregate)

Returns the concatenation string of the specified expression.

Syntax

```
STRING_AGG( <expression>[, <delimiter> ] [ <order_by_clause> ] )
```

Syntax Elements

**expression**

Specifies a VARCHAR or NVARCHAR expression with values to be concatenated. If the input values are a different data type than VARCHAR or NVARCHAR, then implicit casting is attempted.

For example, if the NUM column has five integer values (1, 2, 3, 4, 5), then STRING_AGG("NUM",0) returns 102030405.

**delimiter**

Specifies the character to use as a delimiter when aggregating values.

**order_by_clause**

Specifies the sort order of the input rows.

```
<order_by_clause> ::= ORDER BY <order_by_expression> [, <order_by_expression> [, ...] ]
<order_by_expression> ::= <column_name> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [ <collate_clause> ]
                      | <column_position> [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [ <collate_clause> ]
<collate_clause> ::= COLLATE <collation_name>
```

**<collate_clause>** specifies the collation to use for ordering values in the results. **<collate_clause>** can only be used on columns defined as NVARCHAR or VARCHAR. **<collation_name>** is one of the supported collation names listed in the COLLATIONS system view.

Description

NULL values are treated as empty strings.

The default ordering is ASC NULLS FIRST. If DESC is specified, then the ORDER BY expression becomes DESC NULLS LAST.
Example

The example below creates table r1 and populate it with data.

```sql
CREATE ROW TABLE r1(a INT, str VARCHAR(20), grp INT);
INSERT INTO r1 VALUES (3,'str2',0);
INSERT INTO r1 VALUES (0,'str1',0);
INSERT INTO r1 VALUES (NULL,'NULL',0);
INSERT INTO r1 VALUES (5,'str3',0);
INSERT INTO r1 VALUES (3,'val3',1);
INSERT INTO r1 VALUES (6,'val6',1);
INSERT INTO r1 VALUES (NULL,'NULL',1);
INSERT INTO r1 VALUES (1,'val1',1);
```

Execute the following statement to return the concatenation string of each record from table r1 in ascending order.

```sql
SELECT grp, STRING_AGG(str,','ORDER BY a)AS agg FROM r1 GROUP BY grp;
```

The statement above returns the following results.

<table>
<thead>
<tr>
<th>GRP</th>
<th>AGG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NULL,str1,str2,str3</td>
</tr>
<tr>
<td>1</td>
<td>NULL,val1,val3,val6</td>
</tr>
</tbody>
</table>

Execute the following statement to return the concatenation string of each record from table r1 in descending order.

```sql
SELECT grp, STRING_AGG(str,','ORDER BY a DESC) AS agg FROM r1 GROUP BY grp;
```

The statement above returns the following results.

<table>
<thead>
<tr>
<th>GRP</th>
<th>AGG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>str3,str2,str1,NULL</td>
</tr>
<tr>
<td>1</td>
<td>val6,val3,val1,NULL</td>
</tr>
</tbody>
</table>

Related Information

Aggregate Functions [page 414]
COLLATIONS System View [page 1520]
4.9.1.188 STRTOBIN Function (String)

Converts all characters in a string into a binary encoding using the specified codepage.

**Syntax**

```
STRTOBIN(<string>, <codepage>)
```

**Description**

Converts all characters in string `<string>` into a binary encoding using the defined `<codepage>`.

**Example**

This example converts all characters in the given string to binary UTF-16BE encoding, and returns the value 0041006E0074:

```
SELECT STRTOBIN ('Ant', 'UTF-16BE') "strtobin" FROM DUMMY;
```

**Related Information**

Character String Data Types [page 36]

4.9.1.189 SUBARRAY Function (Array)

Returns a subset of values from the specified array beginning from the specified start position.

**Syntax**

```
SUBARRAY(<array_value_expression>, <start_position> , <length>)
```
Description

array_value_expression Specifies the array that the function returns the subset of values for.
start_position Specifies where in the array the subset of values begins.
length Determines the number of values.

Returns a values set of <array_value_expression> starting from the <start_position> within the string.

If <start_position> is less than or equal to 0, then it is considered as 1.
If the <length> is less than or equal to 0, or it is greater than the remaining part of <array_value_expression>, then SUBARRAY returns the remaining part from the <start_position>.

Example

The following example returns subsets from the specified array:

```sql
CREATE COLUMN TABLE ARRAY_TEST (IDX INT, VAL INT ARRAY);
INSERT INTO ARRAY_TEST VALUES (1, ARRAY(1, 2, 3));
INSERT INTO ARRAY_TEST VALUES (2, ARRAY(10, 20, 30, 40));
```

The example below generates a subarray that begins with 1 and has a length of 2:

```sql
SELECT SUBARRAY(VAL, 1, 2) "subarray" FROM ARRAY_TEST;
```

The statement above returns the following results:

<table>
<thead>
<tr>
<th>Subarray</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2</td>
</tr>
<tr>
<td>10, 20</td>
</tr>
<tr>
<td>1, 2</td>
</tr>
<tr>
<td>10, 20</td>
</tr>
</tbody>
</table>

The example below generates a subarray beginning with 1 and with a length of 10:

```sql
SELECT SUBARRAY(VAL, 1, 10) "subarray" FROM ARRAY_TEST;
```

The statement above returns the following results:

<table>
<thead>
<tr>
<th>Subarray</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2, 3</td>
</tr>
<tr>
<td>10, 20, 30, 40</td>
</tr>
</tbody>
</table>
4.9.1.190 SUBSTR_AFTER Function (String)

Returns a substring from a specified string that follows the first occurrence of the specified pattern.

Syntax

```
SUBSTR_AFTER(<string>, <pattern>)
```

Syntax Elements

- **string**: Specifies the string containing the substring.
- **pattern**: Specifies the occurrence within the string that marks the place immediately before the returned substring.

Description

Returns a substring from the specified string that follows the first occurrence of the specified pattern.

- If `<string>` does not contain the `<pattern>` substring, then an empty string is returned.
- If `<pattern>` is an empty string, then `<string>` is returned.
- If `<string>` or `<pattern>` is NULL, then NULL is returned.
Example

This example returns Friend, the part of the given string that is to the right of the first occurrence of My:

```
SELECT SUBSTR_AFTER ('Hello My Friend','My ') "substr after" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]

4.9.1.191  SUBSTR_BEFORE Function (String)

Returns a substring from a specified string before the first occurrence of the specified pattern.

Syntax

```
SUBSTR_BEFORE(<string>, <pattern>)
```

Syntax Elements

- **string** Specifies the string containing the substring.
- **pattern** Specifies the occurrence within the string that marks the place immediately after the returned substring.

Description

Returns a substring from the specified string before the first occurrence of the specified pattern.

- If `<string>` does not contain the `<pattern>` substring, then an empty string is returned.
- If `<pattern>` is an empty string, then `<string>` is returned.
- If `<string>` or `<pattern>` is NULL, then NULL is returned.
Example

The following example returns Hello, the part of the given string that is to the left of the first occurrence of My:

```
SELECT SUBSTR_BEFORE ('Hello My Friend','My') "substr before" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]

### 4.9.1.192 SUBSTRING_REGEXPR Function (String)

Searches a string for a regular expression pattern and returns one occurrence of the matching substring.

Syntax

```
SUBSTR[ING]_REGEXPR( <pattern> [ FLAG <flag> ] IN <regex_subject_string>
    [ FROM <start_position> ]
    [ OCCURRENCE <regex_occurrence> ]
    [ GROUP <regex_capture_group> ] )
```

Syntax Elements

- **pattern**: Specifies a search pattern based on Perl Compatible Regular Expression (PCRE).
  
  ```
  <pattern> ::= !!Perl Compatible Regular Expression
  ```

- **flag**: Specifies that the matching behavior of the function can be defined by the flag literal. The following options are available:

<table>
<thead>
<tr>
<th>Flag option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>Enables case-insensitive matching</td>
</tr>
<tr>
<td>m</td>
<td>Enables multiline mode, where the <code>&lt;subject_string&gt;</code> will be treated as multiple lines and the expression <code>^</code> and <code>$</code> match just after or just before, respectively, a line terminator or the end of the input sequence</td>
</tr>
<tr>
<td>s</td>
<td>Enables the expression <code>&lt;&gt;</code> as a wildcard to match any character, including a line terminator</td>
</tr>
<tr>
<td>x</td>
<td>Permits whitespace and comments in the pattern</td>
</tr>
</tbody>
</table>
SAP HANA uses the Perl-compatible Regular Expressions (PCRE) library to process regular expressions. Specifying 'U' (short for "ungreedy") inverts the "greediness" of quantifiers so that they are not greedy by default but become greedy only when followed by "?". Ungreedy matching can often perform faster because it finds the shorter match at times when it is only interesting to know whether there is any match.

For a full understanding of the "greedy" versus "ungreedy" matching behavior of the Perl-compatible Regular Expressions (PCRE) library, visit: https://www.pcre.org/original/doc/html/pcre-matching.html

<flag> ::= 'i' | 'm' | 's' | 'x'

regex_subject_string Defines the string that the search pattern should be applied to. If <regex_subject_string> is empty, then the result is empty.

<regex_subject_string> ::= <string>

start_position Setting this parameter to a positive integer indicates which character of <regex_subject_string> the search begins at. If <start_position> is not a positive integer, then NULL is returned.

<start_position> ::= <numeric_literal>

regex_occurrence Setting this parameter to a positive integer indicates the occurrence of the <pattern> in <regex_subject_string>. The default is 1 and returns the first occurrence. If <regex_occurrence> is not a positive integer, then NULL is returned.

<regex_occurrence> ::= <numeric_literal >

regex_capture_group Indicates the number of the captured substring's group by the regular expression. The default is 0. If <regex_capture_group> is a negative integer, then 0 is returned.

<regex_capture_group> ::= <integer>

**Description**

Searches a string for a regular expression pattern and returns one occurrence of the matching substring.

If any of the following parameters is NULL, then the function returns NULL: <pattern>, <flag>, <regex_subject_string>, <start_position>, <regex_occurrence>, or <regex_capture_group>.

**Example**

The following example returns the day 01 from the date value 20140401:

```sql
SELECT SUBSTR_REGEXPR('([[:digit:]]{4})([[:digit:]]{2})([[:digit:]]{2})' IN '20140401' GROUP 3) "substring_regexpr" FROM DUMMY;
```
4.9.1.193 SUBSTRING Function (String)

Returns a substring from an input value, starting from a specified position within the input value.

Syntax

```
SUBSTRING(<string>, <start_position> [, <string_length>])
```

Syntax Elements

- **string**: Specifies the string containing the substring. `<string>` can also be a binary type BLOB or VARBINARY.
- **start_position**: Specifies where the substring starts within the string.
- **string_length**: Specifies the substring length.

Description

Returns a substring from the specified string starting from `<start_position>` within the string. `SUBSTRING` can return the remaining part of a string from the `<start_position>`, or optionally, a number of characters set by the `<string_length>` parameter.

- If `<start_position>` is less than 1, then it is considered to be 1.
- If `<string_length>` is less than 1, then an empty string is returned.
- If `<string_length>` is greater than the length of remaining part of `<str>`, then the remaining part is returned without blank padding.

When used on binary types, `SUBSTRING` factors in byte length and interprets the offsets, `<start_position>` and `<string_length>`, as byte positions.
Example

The following example selects two characters from the string `1234567890` starting at position 4, and returns the value 45:

```
SELECT SUBSTRING ('1234567890', 4, 2) "substring" FROM DUMMY;
```

The following example returns 'ABCD' from the binary value `x'ABCDEF'`:

```
SELECT SUBSTRING(x'ABCDEF', 1, 2) "substring" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]

4.9.1.194  SUM Function (Aggregate)

Returns the sum of the expression. This function can also be used as a window function.

Syntax

Aggregate function:

```
SUM( [ ALL | DISTINCT ] <expression> )
```

Window function:

```
SUM( <expression> ) <window_specification>
```

Syntax Elements

expression

Specifies the input data for the function.

window_specification

Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].
Description

Result type based on input

<table>
<thead>
<tr>
<th>TINYINT</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>BIGINT</th>
<th>DECIMAL (p, s)</th>
<th>DECIMAL</th>
<th>REAL</th>
<th>DOUBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
<td>INTEGER</td>
<td>BIGINT</td>
<td>DECIMAL(p, s)</td>
<td>DECIMAL</td>
<td>REAL</td>
<td>DOUBLE</td>
</tr>
</tbody>
</table>

Example

The following statements create a table with data for example purposes:

```
DROP TABLE "MyProducts";
CREATE TABLE "MyProducts"
    ("Product_ID" VARCHAR(10),
     "Product_Name" VARCHAR(100),
     "Category" VARCHAR(100),
     "Quantity" INTEGER,
     "Price" DECIMAL(10,2),
     PRIMARY KEY ("Product_ID"));
```

```
INSERT INTO "MyProducts" VALUES('P1','Shirts', 'Clothes', 32, 20.99);
INSERT INTO "MyProducts" VALUES('P2','Jackets', 'Clothes', 16, 99.49);
INSERT INTO "MyProducts" VALUES('P3','Trousers', 'Clothes', 30, 32.99);
INSERT INTO "MyProducts" VALUES('P4','Coats', 'Clothes', 5, 129.99);
INSERT INTO "MyProducts" VALUES('P5','Purse', 'Accessories', 3, 89.49);
```

The following example sums the quantities of coats and jackets, and returns the value 21:

```
SELECT SUM("Quantity") FROM "MyProducts" WHERE "Product_Name" IN ('Jackets', 'Coats');
```

Related Information

- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- Aggregate Functions [page 414]
- Expressions [page 56]
4.9.1.195  SYSUUID Function (Miscellaneous)

Returns a new universally unique identifier that is generated by the connected SAP HANA instance.

Syntax

```sql
SYSUUID
```

Description

Each time you call the SYSUUID function, it returns a new UUID value. SYSUUID calls from multiple connections are internally serialized to guarantee unique value generation.

Example

The following query returns the uniquely generated value `CF773C584E3A6213E2008AC320F0F2BE`:

```sql
SELECT SYSUUID FROM DUMMY;
```

4.9.1.196  TAN Function (Numeric)

Returns the tangent of a specified number, where the argument is an angle expressed in radians.

Syntax

```sql
TAN(<number>)
```

Description

Returns the tangent of `<number>`, where `<n>` is an angle expressed in radians.
Example

The following example returns the value 0 for "tan":

```
SELECT TAN (0.0) "tan" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.197 TANH Function (Numeric)

Returns the hyperbolic tangent of the specified numeric argument.

Syntax

```
TANH(<number>)
```

Description

Returns the hyperbolic tangent of the numeric argument `<number>`.

Example

The following example returns the value 0.7615941559557649 for "tanh":

```
SELECT TANH (1.0) "tanh" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]
4.9.1.198  TO_ALPHANUM Function (Data Type Conversion)

Converts a given value to an ALPHANUM data type.

Syntax

```
TO_ALPHANUM(<value>)
```

Description

Converts a given `<value>` to an ALPHANUM data type.

Example

The following example converts the value `10` to the ALPHANUM value `10`.

```
SELECT TO_ALPHANUM ('10') "to alphanum" FROM DUMMY;
```

Related Information

Data Type Conversion [page 37]

4.9.1.199  TO_BIGINT Function (Data Type Conversion)

Converts a value to a BIGINT data type.

Syntax

```
TO_BIGINT(<value>)
```
Description

If the input `<value>` has a mantissa, then these digits are truncated during the conversion process.

Examples

The following example converts the value 10 to a BIGINT value 10:

```
SELECT TO_BIGINT ('10') "to bigint" FROM DUMMY;
```

The following example converts the value 10.5 to a BIGINT value 10, truncating the mantissa:

```
SELECT TO_BIGINT (10.5) "to bigint" FROM DUMMY;
```

Related Information

Data Type Conversion [page 37]

4.9.1.200  TO_BINARY Function (Data Type Conversion)

Converts a value to a BINARY data type.

Syntax

```
TO_BINARY(<value>)
```

Description

Converts a `<value>` to a BINARY data type.

Example

The following example converts the value abc to the BINARY value 616263.

```
SELECT TO_BINARY ('abc') "to binary" FROM DUMMY;
```
4.9.1.201 TO_BLOB Function (Data Type Conversion)

Converts a binary string or (N)CLOB data type to a BLOB data type.

**Syntax**

```sql
TO_BLOB(<value>)
<value> ::= <binary_string> | <nclob_value> | <clob_value>
```

**Description**

Converts a BINARY, CLOB, or NCLOB `<value>` to a BLOB data type.

**Example**

The following example converts the BINARY value `abcde` to the BLOB value `abcde`:

```
SELECT TO_BLOB (TO_BINARY('abcde')) "to blob" FROM DUMMY;
```

The following example converts the CLOB value `abc` to the BLOB value `abc`:

```
SELECT TO_BLOB (TO_CLOB('abc')) "to blob" FROM DUMMY;
```
4.9.1.202  TO_BOOLEAN Function (Data Type Conversion)

Converts a value to a BOOLEAN data type.

Syntax

TO_BOOLEAN(<value>)

Syntax Elements

value if <value> is:

- 1, 'true', or true then TO_BOOLEAN returns the value true.
- 0, 'false', or false then TO_BOOLEAN returns the value false.
- unknown (Boolean) or 'unknown' then TO_BOOLEAN returns the value unknown.

'true', 'false', and 'unknown' are not case sensitive, meaning that values of 'TRUE' and 'true' are treated the same as a value of true. TO_BOOLEAN throws an exception when <value> is not one of true, false, unknown, 0, 1, '0', '1', 'true', 'false' or 'unknown'.

Description

Converts a <value> to a BOOLEAN data type.

Example

Converts the value 0 to the BOOLEAN value FALSE.

```
SELECT TO_BOOLEAN(0) FROM DUMMY;
```

Converts the value 1 to the BOOLEAN value TRUE.

```
SELECT TO_BOOLEAN(1) FROM DUMMY;
```

Related Information

Boolean Data Type [page 35]
4.9.1.203  TO_CLOB Function (Data Type Conversion)

Converts a value to a CLOB data type.

Syntax

TO_CLOB(<value>)

Description

Converts a <value> to a CLOB data type.

Example

The following example converts the value TO_CLOB converts the value to a CLOB data type to a CLOB value.

SELECT TO_CLOB ('TO_CLOB converts the value to a CLOB data type') "to clob" FROM DUMMY;

Related Information

Data Type Conversion [page 37]

4.9.1.204  TO_DATE Function (Data Type Conversion)

Converts a date string into a DATE data type.

Syntax

TO_DATE(<date> [, <format>])

Description

Converts the date string `<date>` into a DATE data type.
If the `<format>` specifier is omitted, the conversion is performed using the date format model.

Example

The following example converts the string `2010-01-12` to a DATE value with the format `YYYY-MM-DD`, and returns the value `2010-01-12` (or another format like `Jan 12, 2010`, depending on your date display settings):

```
SELECT TO_DATE('2010-01-12', 'YYYY-MM-DD') "to date" FROM DUMMY;
```

Related Information

Data Type Conversion [page 37]

4.9.1.205 TO_DATS Function (Data Type Conversion)

Converts a date string into an ABAP DATE string.

Syntax

```
TO_DATS(<date>)
```

Description

Converts the date string `<date>` into an ABAP DATE string with format `YYYYMMDD`.

Example

The following example converts the value `2010-01-12` to the ABAP DATE string `20100112`:

```
SELECT TO_DATS ('2010-01-12') "abap date" FROM DUMMY;
```
Related Information

Data Type Conversion [page 37]

4.9.1.206  TO_DECIMAL Function (Data Type Conversion)

Converts a value to a DECIMAL data type.

Syntax

```
TO_DECIMAL(<value> [, <precision>, <scale>])
```

Syntax Elements

- **value**: Specifies the value to convert to a DECIMAL data type. `<value>` can be a variable.
- **precision**: Specifies the total number of significant digits. `<precision>` can range from 1 to 38, specified as a string constant. If `<precision>` is not specified, 34 is the default.
- **scale**: Specifies the number of digits from the decimal point to the least significant digit. `<scale>`. `<scale>` must be between 0 and the value for `<precision>`. The scale is positive when the number has significant digits to the right of the decimal point, and negative when the number has significant digits to the left of the decimal point.

Description

If scale is not specified, then the default is 0. If both `<precision>` and `<scale>` are not specified, then DECIMAL becomes a floating-point decimal number, covering 1 to 34 for precision, and -6,111 to 6,176 for scale, depending on `<value>`.

Unnecessary least significant digits in the mantissa of the input value are truncated during the conversion process.

Example

The following example converts the value `7654321.888888` to DECIMAL with 10 digits precision and a scale of 3, and returns the value `7654321.888`:

```
SELECT TO_DECIMAL(7654321.888888, 10, 3) "to decimal" FROM DUMMY;
```
The following example converts the value 1234.789 to DECIMAL with 10 for precision and 0 for scale, and returns the value 1234 (scale is defined as 0, so everything is truncated after precision).

```
SELECT TO_DECIMAL(1234.789,10,0) AS "to decimal" FROM DUMMY;
```

The following example, converts the value 1234567890123456789012345678901256.78 to a DECIMAL data type, with 34 as precision (the default) and 0 as the scale (the default), and returns 1234567890123456789012345678901256 (the last two digits are truncated because the default precision is 34).

```
SELECT TO_DECIMAL(1234567890123456789012345678901256.78) "to decimal" FROM DUMMY;
```

Related Information

- Data Type Conversion [page 37]
- Numeric Data Types [page 48]

4.9.1.207 TO_DOUBLE Function (Data Type Conversion)

Converts a value to a DOUBLE data type.

**Syntax**

```
TO_DOUBLE(<value>)
```

**Description**

Converts a specified `<value>` to a DOUBLE (double precision) data type.

**Example**

The following example converts 15.12 to a DOUBLE, and then multiplies the value by 3, returning the DOUBLE value 45.36.

```
SELECT 3*TO_DOUBLE ('15.12') "to double" FROM DUMMY;
```
Related Information

Data Type Conversion [page 37]

4.9.1.208  TO_FIXEDCHAR Function (Data Type Conversion)

Converts a specified number of characters from a string starting at the first character in the string.

Syntax

```
TO_FIXEDCHAR(<string>, <size>)
```

Description

Converts the <string> to the specified <size>, starting at the first character. <size> cannot be a variable.

Example

The following example converts the value Ant to a length of 2, and returns the value An.

```
SELECT TO_FIXEDCHAR ('Ant', 2) "to_fixedchar" FROM DUMMY;
```

Related Information

Data Type Conversion [page 37]
4.9.1.209 TO_INT Function (Data Type Conversion)

Converts a value to an INT data type.

**Syntax**

```
TO_INT(<value>)
```

**Description**

If the input `<value>` has a mantissa, then the mantissa is truncated during the conversion process.

**Examples**

The following example converts the value 10 to the INT value 10:

```
SELECT TO_INT('10') "to int" FROM DUMMY;
```

The following example converts the value 10.5 to the INT value 10, truncating the mantissa:

```
SELECT TO_INT(10.5) "to int" FROM DUMMY;
```

**Related Information**

Data Type Conversion [page 37]

4.9.1.210 TO_INTEGER Function (Data Type Conversion)

Converts the `<value>` to an INTEGER data type.

**Syntax**

```
TO_INTEGER(<value>)
```
Description

If the input `<value>` has a mantissa, then these digits are truncated during the conversion process.

Examples

The following example converts the value `10` to the INTEGER value `10`:

```sql
SELECT TO_INTEGER ('10') "to int" FROM DUMMY;
```

The following example converts the value `10.5` to the INTEGER value `10`, truncating the mantissa:

```sql
SELECT TO_INTEGER(10.5) "to int" FROM DUMMY;
```

Related Information

Data Type Conversion [page 37]

4.9.1.211 TO_JSON_BOOLEAN (Data Type Conversion)

Converts a given `<value>` to a boolean value in JSON format.

Syntax

```sql
TO_JSON_BOOLEAN(<value>)
```

Description

If the `<value>` cannot be converted to a JSON boolean value, then an error is returned.

Example

Create and populate the collection `myCollection1`:

```sql
CREATE COLLECTION myCollection1;
```
INSERT INTO myCollection1 VALUES ("{"k1" : true}");

Running the following statement returns an error:

SELECT * FROM myCollection1 WHERE "k1" = TRUE;

Running the following statement returns the record:

SELECT * FROM myCollection1 WHERE "k1" = TO_JSON_BOOLEAN(TRUE);

### 4.9.1.212 TO_NCLOB Function (Data Type Conversion)

Converts a `<value>` to an NCLOB data type.

**Syntax**

```
TO_NCLOB(<value>)
```

**Description**

Converts a `<value>` to an NCLOB data type.

**Example**

The following example converts the value `TO_NCLOB converts the value to a NCLOB data type` to an NCLOB value.

```
SELECT TO_NCLOB ('TO_NCLOB converts the value to a NCLOB data type') "to nclob"
FROM DUMMY;
```

**Related Information**

Data Type Conversion [page 37]
**4.9.1.213  TO_NVARCHAR Function (Data Type Conversion)**

Converts a given value to an NVARCHAR data type, with an option to format the output value.

**Syntax**

```
TO_NVARCHAR(<value> [, <format>])
```

**Syntax Elements**

- **format**
  
  Specifies the desired formatting to apply to the converted value. For datetime values, use `<format>` to define the desired datetime output format (for example, `YYYY/MM/DD`). See the Datetime Data Types topic in this guide for more information on possible datetime formats.

  To define other types of output formats, for example numbers with custom symbols, commas, zeros and so on, specify `<format>` using the following conventions:
  
  - 9 - Return the number in the specified position; otherwise, return nothing.
  - 0 - Return the number in the specified position; otherwise, return a zero (0).
  - S - Return the sign symbol (either + or -) for the value.
  - E - Divide the number into significant part and exponent part.
  - % - Multiply `<value>` by 10^2 and adds a percent symbol (%) at the end.
  - . (a period) - Insert a period in the specified position.
  - All other characters other than the items above: Return the character in the specified position.

**Description**

If the `<format>` specifier is omitted, then the conversion is performed using the date format model.

The following data types can be converted to NVARCHAR using the TO_NVARCHAR function:

- ALPHANUM, BIGINT, DATE, DECIMAL, DOUBLE, FIXED12, FIXED16, FIXED8, INTEGER, REAL, SECONDDATE, SMALLDECIMAL, SMALLINT, TIME, TIMESTAMP, TINYINT, VARBINARY, VARCHAR
- CLOB, NCLOB, TEXT (if the value is longer than the maximum length of NVARCHAR, then an exception is thrown)
Example

The following example converts the value **2009/12/31** to the NVARCHAR value **09-12-31**.

```sql
SELECT TO_NVARCHAR(TO_DATE('2009/12/31'), 'YY-MM-DD') "to nvarchar" FROM DUMMY;
```

The following statements demonstrate how to use the `<string_formatting>` option to define the format of the output values:

```sql
SELECT TO_NVARCHAR(1, '00.00') FROM Dummy;   --> 01.00
SELECT TO_NVARCHAR(100, '00.00') FROM Dummy;   --> 100.00
SELECT TO_NVARCHAR(100, '9999.00') FROM Dummy; --> 100.00
SELECT TO_NVARCHAR(100, '0000.00') FROM Dummy; --> 0100.00
SELECT TO_NVARCHAR(100, 'S0000.00') FROM Dummy; --> +0100.00
SELECT TO_NVARCHAR(-100, 'S0000.00') FROM Dummy; --> -0100.00
SELECT TO_NVARCHAR(-100, '000.00') FROM Dummy; --> -100.00
SELECT TO_NVARCHAR(-100, 'S0.0E0') FROM Dummy; --> -1.0E2
SELECT TO_NVARCHAR(-0.001, 'S0.0E00') FROM Dummy; --> -1.0E-3
SELECT TO_NVARCHAR(-0.001, 'S0.0E00') FROM Dummy; --> -1.0E-03
SELECT TO_NVARCHAR(1000, '9,999.00') FROM Dummy; --> 1,000.00
SELECT TO_NVARCHAR(1000, '$9,999.00') FROM Dummy; --> $1,000.00
SELECT TO_NVARCHAR(1000, '$9,999.99') FROM Dummy; --> $1,000.
SELECT TO_NVARCHAR(1, '0.00%') FROM Dummy;     --> 100.00%
SELECT TO_NVARCHAR(1, '100%') FROM dummy;       --> 1100%
```

Related Information

- Datetime Data Types [page 40]
- TO_VARCHAR Function (Data Type Conversion) [page 383]
- Data Type Conversion [page 37]

### 4.9.1.214 TO_REAL Function (Data Type Conversion)

Converts a `<value>` to a REAL data type.

**Syntax**

```
TO_REAL(<value>)
```

**Description**

Converts a `<value>` to a REAL (single precision) data type.
Example

The following example converts the value 15.12 to a REAL value and multiplies it by 3 to return the value 45.36000061035156.

```
SELECT 3*TO_REAL ('15.12') "to real" FROM DUMMY;
```

Related Information

Data Type Conversion [page 37]

4.9.1.215 TO_SECONDDATE Function (Data Type Conversion)

Converts a specified date string to a SECONDDATE data type.

Syntax

```
TO_SECONDDATE(<date> [, <format>])
```

Description

If the <format> specifier is omitted, then the conversion is performed using the date format model.

Example

The following example converts the value 2010-01-11 13:30:00 to a SECONDDATE data type with format YYYY-MM-DD HH24:MI:SS and returns the value 2010-01-11 13:30:00.0 (or another format like Jan 11, 2010 1:30:00.0 PM, depending on your date display settings):

```
SELECT TO_SECONDDATE ('2010-01-11 13:30:00', 'YYYY-MM-DD HH24:MI:SS') "to seconddate" FROM DUMMY;
```
4.9.1.216  TO_SMALLDECIMAL Function (Data Type Conversion)

Converts the specified value to a SMALLDECIMAL data type.

Syntax

```
TO_SMALLDECIMAL(<value>)
```

Description

Converts the specified value to a SMALLDECIMAL data type.

Example

The following example converts the value `7654321.89` to the SMALLDECIMAL value `7,654,321.89`:

```
SELECT TO_SMALLDECIMAL(7654321.89) "to smalldecimal" FROM DUMMY;
```

Related Information

Data Type Conversion [page 37]
4.9.1.217  TO_SMALLINT Function (Data Type Conversion)

Converts a value to a SMALLINT data type.

**Syntax**

```
TO_SMALLINT(<value>)
```

**Description**

If the input `<value>` has a mantissa, then these digits are truncated during the conversion process.

**Examples**

The following example converts the value 10 to a SMALLINT and returns the value 10:

```
SELECT TO_SMALLINT ('10') "to smallint" FROM DUMMY;
```

The following example converts the value 10 to a SMALLINT and returns the value 10, truncating the mantissa:

```
SELECT TO_SMALLINT(10.5) "to int" FROM DUMMY;
```

**Related Information**

Data Type Conversion [page 37]

4.9.1.218  TO_TIME Function (Data Type Conversion)

Converts a specified time string to a TIME data type.

**Syntax**

```
TO_TIME(<time> [, <format>])
```
Description

Converts time string `<time>` to the TIME data type.

If the `<format>` specifier is omitted, then the conversion is performed using the time format model.

Example

The following example converts the value `08:30 AM` to a TIME value with format `HH:MI AM` and returns the value `08:30:00`:

```
SELECT TO_TIME ('08:30 AM', 'HH:MI AM') "to time" FROM DUMMY;
```

Related Information

Data Type Conversion [page 37]

4.9.1.219  TO_TIMESTAMP Function (Data Type Conversion)

Converts a date string to a TIMESTAMP data type.

Syntax

```
TO_TIMESTAMP (<date> [, <format>])
```

Description

Converts date string `<date>` to the TIMESTAMP data type.

If the `<format>` specifier is omitted, then the conversion is performed using the date format model.
Example

The following example converts the value 2010-01-11 13:30:00 to the TIMESTAMP value 2010-01-11 13:30:00.0 using the format YYYY-MM-DD HH24:MI:SS:

```
SELECT TO_TIMESTAMP ('2010-01-11 13:30:00', 'YYYY-MM-DD HH24:MI:SS') "to timestamp" FROM DUMMY;
```

Related Information

Data Type Conversion [page 37]

4.9.1.220 TO_TINYINT Function (Data Type Conversion)

Converts a value to a TINYINT data type.

Syntax

```
TO_TINYINT(<value>)
```

Description

Converts the `<value>` to a TINYINT data type.

If the input value has a mantissa, then these digits are truncated during the conversion process.

Examples

The following example converts the value 10 to the TINYINT value 10:

```
SELECT TO_TINYINT ('10') "to tinyint" FROM DUMMY;
```

The following example converts the value 10.5 to the TINYINT value 10, truncating the mantissa:

```
SELECT TO_TINYINT(10.5) "to tinyint" FROM DUMMY;
```
4.9.1.221 TO_VARCHAR Function (Data Type Conversion)

In SAP HANA, VARCHAR is an alias for the NVARCHAR data type. Please use the TO_NVARCHAR function instead.

Syntax

TO_VARCHAR(<value> [, <format>])

Syntax Elements

value

The input value for the function. If <value> is a datetime data type, and <format> is not specified, then the default format is applied to the output value, unless the relevant session variable (DATE_FORMAT, TIME_FORMAT, TIMESTAMP_FORMAT, or SECONDDATE_FORMAT) is set to something different.

format

Specifies the desired formatting to apply to the converted value. For datetime values, use <format> to define the desired datetime output format (for example, YYYY/MM/DD. See the Datetime Data Types topic in this guide for more information on possible datetime formats.

To define other types of output formats, for example numbers with custom symbols, commas, zeros and so on, specify <format> using the following conventions:

• 9 - Return the number in the specified position; otherwise, return nothing.
• 0 - Return the number in the specified position; otherwise, return a zero (0).
• S - Return the sign symbol (either + or -) for the value.
• E - Divide the number into significant part and exponent part.
• % - Multiply <value> by 10^2 and adds a percent symbol (%) at the end.
• . (a period) - Insert a period in the specified position.
• All other characters other than the items above: Return the character in the specified position.
Description

Converts a given value to a VARCHAR data type, with an option to format the output value. While the function is called TO_VARCHAR, in SAP HANA, VARCHAR is an alias for the NVARCHAR data type. Therefore, please use the TO_NVARCHAR function instead.

Example

The following example converts the value 2009-12-31 to a date value with the format YYYY/MM/DD. It then converts it again to an NVARCHAR type and returns the value 2009/12/31.

```sql
SELECT TO_VARCHAR (TO_DATE('2009-12-31'), 'YYYY/MM/DD') "to varchar" FROM DUMMY;
```

The following statements demonstrate how to use the `<string_formatting>` option to define the format of the output values:

```sql
SELECT TO_VARCHAR(1, '00.00') FROM Dummy;    --> 01.00
SELECT TO_VARCHAR(100, '00.00') FROM Dummy;   --> 100.00
SELECT TO_VARCHAR(100, '9999.00') FROM Dummy; --> 100.00
SELECT TO_VARCHAR(100, '0000.00') FROM Dummy;  --> 100.00
SELECT TO_VARCHAR(100, '0000.00') FROM Dummy;  --> 00100
SELECT TO_VARCHAR(100, '9999.99') FROM Dummy;  --> 100.00
SELECT TO_VARCHAR(100, '9999.99') FROM Dummy;  --> 100.00
SELECT TO_VARCHAR(1,'0.00%') FROM Dummy;      --> 100.00%
SELECT TO_VARCHAR(1,'100%') FROM Dummy;        --> 1100%
```

Related Information

Datetime Data Types [page 40]
TO_NVARCHAR Function (Data Type Conversion) [page 376]
Data Type Conversion [page 37]
4.9.1.222 TRIM Function (String)

Returns a string after removing leading and trailing spaces.

Syntax

\[
\text{TRIM([LEADING | TRAILING | BOTH] <trim_char> FROM} \ <\text{string}> \ )
\]

Syntax Elements

\[\text{trim_char}\]

Specifies the characters to be removed from a string.

Description

Returns string \(<\text{string}>\) after removing leading and trailing spaces. The trimming operation is carried out either from the start (LEADING), end (TRAILING) or both (BOTH) ends of the string.

- If either \(<\text{string}>\) or \(<\text{trim_char}>\) are NULL values, then NULL is returned.
- If no options are specified, TRIM removes both the leading and trailing substring \(<\text{trim_char}>\) from string \(<\text{string}>\).
- If \(<\text{trim_char}>\) is not specified, then a single blank space is used.

Example

The following example removes the character \(a\) both at the beginning and at the end of the specified string, and returns the value 123456789:

\[
\text{SELECT TRIM ('a' FROM 'aaa123456789aa') "trim both" FROM DUMMY;}
\]

The following example removes the character \(a\) at the begin of the specified string, and returns the value 123456789aa:

\[
\text{SELECT TRIM (LEADING 'a' FROM 'aaa123456789aa') "trim leading" FROM DUMMY;}
\]
4.9.1.223 TRIM_ARRAY Function (Array)

Removes the specified number of elements from the end of an array.

**Syntax**

```
TRIM_ARRAY(<array_value_expression>, <truncate_length>)
```

**Syntax Elements**

- **array_value_expression** Specifies the array that the function removes the elements from.
- **truncate_length** Specifies the number of elements to remove at the end of the array.

**Description**

Returns an array that has a specified number of elements trimmed from the end.

If `<truncate_length>` is less than or equal to 0, then an exception is thrown.

**Example**

The following example demonstrates the removal of three elements from an array. The query returns 10.

```
CREATE COLUMN TABLE ARRAY_TEST (IDX INT, VAL INT ARRAY);
INSERT INTO ARRAY_TEST VALUES (1, ARRAY(1, 2, 3));
INSERT INTO ARRAY_TEST VALUES (2, ARRAY(10, 20, 30, 40));
SELECT TRIM_ARRAY(VAL, 3) "trim_array" FROM ARRAY_TEST;
```

**Related Information**

Expressions [page 56]
4.9.1.224  UCASE Function (String)

Converts all characters in the specified string to uppercase.

**Syntax**

```
UCASE(<string>)
```

**Description**

Converts all characters in string `<string>` to uppercase.

The UCASE function is identical to the UPPER function.

**Example**

The following example converts the given string to uppercase, and returns the value ANT:

```
SELECT UCASE ('Ant') "ucase" FROM DUMMY;
```

**Related Information**

Character String Data Types [page 36]

4.9.1.225  UMINUS Function (Numeric)

Returns the negated value of the specified numeric argument.

**Syntax**

```
UMINUS(<number>)
```
Description

Returns the negated value of the specified numeric argument.

Example

The following example returns the value 765 for "uminus":

```sql
SELECT UMINUS(-765) "uminus" FROM DUMMY;
```

Related Information

Numeric Data Types [page 48]

4.9.1.226 UNICODE Function (String)

Returns an integer containing the Unicode code point of the first character in the specified string.

Syntax

```sql
UNICODE(<character>)
```

Description

Returns an integer containing the Unicode code point of the first character in the string `<character>`, or NULL if the first character is not a valid encoding.

Example

The following example returns the Unicode code point for the given character (35):

```sql
SELECT UNICODE ('#') "unicode" FROM DUMMY;
```
4.9.1.227 UPPER Function (String)

Converts all characters in a string to uppercase.

Syntax

UPPER(<string>)

Description

Converts all characters in string <string> to uppercase.
The UPPER function is identical to the UCASE function.

Example

This example converts the given string Ant to uppercase and returns the value ANT:

```
SELECT UPPER ('Ant') "uppercase" FROM DUMMY;
```

Related Information

Character String Data Types [page 36]
4.9.1.228  UTCTOLOCAL Function (Datetime)

Converts the specified timestamp between UTC and local time.

Syntax

\[ \text{UTCTOLOCAL} \left( \text{<time>} \ [\text{, <timezone>} [\text{, <timezone_dataset>]}] \right) \]

Syntax Elements

time
Specifies a timestamp in UTC to be converted to local time.
\[ <\text{time}> ::= <\text{timestamp}> \]

timezone
Specifies a string parameter holding the timezone defining the local time. If the local timezone is not explicitly specified, then the local timezone of the SAP HANA system is used.
\[ <\text{timezone}> ::= <\text{string_literal}> \]

timezone_dataset
Specifies the dataset in which to search for the given timezone.
\[ <\text{timezone_dataset}> ::= <\text{string_literal}> \]

Possible values of this string parameter are:

- `sap` searches in the dataset in the timezone definition tables. This is the default value. To use this value, import a timezone dataset (see SAP Note 1791342). If you use this value without importing a dataset, the function uses hardcoded fallback values. These fallback values are deprecated and will be removed in a future support package stack.
- `platform` searches in the dataset provided by the operating system.

Description

Converts the UTC(GMT) time \( t \) to the local time in a timezone.

Use UTC times instead of local timestamps. The use of local times or conversion between local time zones might require additional handling in the application code.
Examples

The following example returns the value Dec 31, 2011 8:00:00.0 PM as the local time equivalent (in EST) of the specified UTC time:

```
SELECT UTCTOLOCAL (TO_TIMESTAMP('2012-01-01 01:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'EST') "utctolocal" FROM DUMMY;
```

The following example returns the value Dec 31, 2011 8:00:00.0 PM as the local time equivalent (in EST) of the specified UTC time:

```
SELECT UTCTOLOCAL (TO_TIMESTAMP('2012-01-01 01:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'EST', 'sap') "utctolocal" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]
SAP Note 1791342

4.9.1.229 VAR Function (Aggregate)

Returns the variance of the given expression as the square of the standard deviation. This function can also be used as a window function.

Syntax

Aggregate function:

```
VAR( [ ALL | DISTINCT ] <expression> )
```

Window function:

```
VAR( <expression> ) <window_specification>
```

Syntax Elements

- `expression`
  - Specifies the input data for the function.
- `window_specification`
Defines a window on the data over which the function operates. For `<window_specification>`, see Window Functions and the Window Specification [page 424].

**Description**

Result type based on input

<table>
<thead>
<tr>
<th></th>
<th>TINYINT</th>
<th>SMALLINT</th>
<th>INTEGER</th>
<th>BIGINT</th>
<th>DECIMAL(p, s)</th>
<th>DECIMAL</th>
<th>REAL</th>
<th>DOUBLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECIMAL</td>
<td>9</td>
<td>DECIMAL</td>
<td>1</td>
<td>DECIMAL</td>
<td>6</td>
<td>5,6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example**

The following statements create a table with data for example purposes:

```sql
DROP TABLE "MyProducts";
CREATE COLUMN TABLE "MyProducts"
  "Product_ID" VARCHAR(10),
  "Product_Name" VARCHAR(100),
  "Category" VARCHAR(100),
  "Quantity" INTEGER,
  "Price" DECIMAL(10,2),
PRIMARY KEY ("Product_ID")
)

INSERT INTO "MyProducts" VALUES('P1','Shirts', 'Clothes', 32, 20.99);
INSERT INTO "MyProducts" VALUES('P2','Jackets', 'Clothes', 16, 99.49);
INSERT INTO "MyProducts" VALUES('P3','Trousers', 'Clothes', 30, 32.99);
INSERT INTO "MyProducts" VALUES('P4','Coats', 'Clothes', 5, 129.99);
INSERT INTO "MyProducts" VALUES('P5','Purse', 'Accessories', 3, 89.49);
```

The following example returns `2128.67`, the variance of values in the `Price` column:

```sql
SELECT VAR("Price") FROM "MyProducts";
```

**Related Information**

- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- Aggregate Functions [page 414]
- VAR_POP Function (Aggregate) [page 393]
- VAR_SAMP Function (Aggregate) [page 394]
4.9.1.230 VAR_POP Function (Aggregate)

Returns the population variance of an expression.

Syntax

VAR_POP(<expression>)

Description

Returns the population variance of the expression as the sum of squares of the difference of <expression> from the mean of <expression>, divided by the number of rows remaining.

Examples

The following example returns 0.2 as the population variance for the specified expression:

```
CREATE ROW TABLE RTABLE (A INT);
INSERT INTO RTABLE VALUES (1);
SELECT VAR_POP(A) "VARPOP" FROM RTABLE;
```

The following example returns 0.24 as the population variance for the specified expression:

```
INSERT INTO RTABLE VALUES (2);
SELECT VAR_POP(A) "VARPOP" FROM RTABLE;
```

Related Information

Window Functions and the Window Specification [page 424]
Window Aggregate Functions [page 428]
Aggregate Functions [page 414]
VAR Function (Aggregate) [page 391]
VAR_SAMP Function (Aggregate) [page 394]
Expressions [page 56]
4.9.1.231 VAR_SAMP Function (Aggregate)

Returns the sample variance of an expression.

Syntax

```
VAR_SAMP(<expression>)
```

Description

Returns the sample variance of the expression as the sum of squares of the difference of `<expression>` from the mean of `<expression>`, divided by the number of rows remaining minus 1. This function is similar to VAR, the only difference is that it returns NULL when the number of rows is 1.

Examples

The following example returns `NULL` as the sample variance of the specified expression:

```
CREATE ROW TABLE RTABLE (A INT);
INSERT INTO RTABLE VALUES (1);
SELECT VAR_SAMP(A) "VARSAMP" FROM RTABLE;
```

The following example returns `0.3` as the sample variance of the specified expression:

```
INSERT INTO RTABLE VALUES (2);
SELECT VAR_SAMP(A) "VARSAMP" FROM RTABLE;
```

Related Information

- Window Functions and the Window Specification [page 424]
- Window Aggregate Functions [page 428]
- Aggregate Functions [page 414]
- VAR Function (Aggregate) [page 391]
- VAR_POP Function (Aggregate) [page 393]
- Expressions [page 56]
4.9.1.232  WEEK Function (Datetime)

Returns a week number between 1 and 54 for the specified date.

Syntax

WEEK(<date>)

Description

Returns the week number of the year as an integer. Weeks start Monday and end Sunday. The first days of the year up to the first Sunday are week 1; the seven days that begin with the day after the first Sunday are week 2, and so on.

Both the WEEK and ISOWEEK functions return the week number for a specified date but the format of the result is quite different, and the two functions may handle the first week of the new year differently. For example, when supplied the date 2017-01-01, the WEEK function considers the date to be part of the first week of 2017 and returns 1, whereas ISOWEEK considers the date to be part of the last week of 2016 and returns 2016-W52.

Example

The following example returns the value 23 for the week number of the specified date:

```sql
SELECT  WEEK (TO_DATE('2011-05-30', 'YYYY-MM-DD')) "week" FROM DUMMY;
```

The following example returns the value 53 for the week number of the specified date:

```sql
SELECT  WEEK ('2016-12-31') FROM DUMMY;
```

The following example returns the value 1 for the week number of the specified date:

```sql
SELECT  WEEK ('2017-01-01') FROM DUMMY;
```

The following example returns the value 2 for the week number of the specified date:

```sql
SELECT  WEEK ('2017-01-02') FROM DUMMY;
```

Related Information

ISOWEEK Function (Datetime) [page 219]
4.9.1.233 WEEKDAY Function (Datetime)

Returns the day of the week for the specified date.

Syntax

WEEKDAY(<date>)

Description

Returns an integer representation of the day of the week for date <date>. The return value ranges from 0 to 6, representing Monday(0) through to Sunday(6).

Example

The following example returns the value 4 as the weekday of the specified date:

```
SELECT WEEKDAY (TO_DATE ('2010-12-31', 'YYYY-MM-DD')) "weekday" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]

4.9.1.234 WEIGHTED_AVG Function (Window)

Computes a weighted moving average by using arithmetically decreasing weights.

Syntax

WEIGHTED_AVG(<expression>) <window_specification>
Syntax Elements

expression

<expression> indicates a numeric column that is used in the weighted moving average calculation.

In a window frame where <n> represents the actual number of elements (ROWS or GROUPS) in the window frame, and <i> represents the element number in the series, the weight factors are calculated with the following algorithm:

\[
\text{weight}(i) = \frac{2 \times (n + 1 - i)}{n \times (n + 1)}
\]

The weight factors are applied in reverse order so that in a window frame with n rows, the first weight factor is multiplied with the value of row n, the second weight factor with the value of row n-1, and so on. When using WEIGHTED_AVG with the GROUPS unit, the values within a group are first arithmetically averaged before they are multiplied with the corresponding weight factor.

window_specification

Defines a window on the data over which the function operates. For <window_specification>, see Window Functions and the Window Specification [page 424].

Description

Computes a weighted moving average by using arithmetically decreasing weights.

Examples

CREATE ROW TABLE weather (station INT, ts DATE, temperature FLOAT);
INSERT INTO weather VALUES(1, '2014-01-01', 0.0);
INSERT INTO weather VALUES(1, '2014-01-02', 3.0);
INSERT INTO weather VALUES(1, '2014-01-03', 4.5);
INSERT INTO weather VALUES(1, '2014-01-04', 6.0);
INSERT INTO weather VALUES(1, '2014-01-05', 6.3);
INSERT INTO weather VALUES(1, '2014-01-06', 5.9);
INSERT INTO weather VALUES(2, '2014-01-01', 1.0);
INSERT INTO weather VALUES(2, '2014-01-02', 3.4);
INSERT INTO weather VALUES(2, '2014-01-03', 5.0);
INSERT INTO weather VALUES(2, '2014-01-04', 6.7);
INSERT INTO weather VALUES(2, '2014-01-05', 4.6);
INSERT INTO weather VALUES(2, '2014-01-06', 6.9);
SELECT ts, temperature, WEIGHTED_AVG(temperature) OVER (ORDER BY ts ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) FROM weather ORDER BY ts;
```
<table>
<thead>
<tr>
<th>ts</th>
<th>TEMPERATURE</th>
<th>WEIGHTED_AVG(temperature) OVER (ORDER BY ts ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) FROM weather ORDER BY ts</th>
</tr>
</thead>
<tbody>
<tr>
<td>01.01.2014</td>
<td>0</td>
<td>0.6666666666666666</td>
</tr>
<tr>
<td>01.01.2014</td>
<td>1</td>
<td>0.6666666666666666</td>
</tr>
<tr>
<td>02.01.2014</td>
<td>3</td>
<td>2.3333333333333335</td>
</tr>
<tr>
<td>02.01.2014</td>
<td>3.4</td>
<td>3.2666666666666666</td>
</tr>
<tr>
<td>03.01.2014</td>
<td>4.5</td>
<td>4.133333333333333</td>
</tr>
<tr>
<td>03.01.2014</td>
<td>5</td>
<td>4.833333333333333</td>
</tr>
<tr>
<td>04.01.2014</td>
<td>6</td>
<td>5.6666666666666666</td>
</tr>
<tr>
<td>04.01.2014</td>
<td>6.7</td>
<td>6.4666666666666667</td>
</tr>
<tr>
<td>05.01.2014</td>
<td>6.3</td>
<td>6.433333333333333</td>
</tr>
<tr>
<td>05.01.2014</td>
<td>4.6</td>
<td>5.1666666666666666</td>
</tr>
<tr>
<td>06.01.2014</td>
<td>5.9</td>
<td>5.4666666666666667</td>
</tr>
<tr>
<td>06.01.2014</td>
<td>6.9</td>
<td>6.5666666666666666</td>
</tr>
</tbody>
</table>
```

**Related Information**

- Expressions [page 56]
- Window Functions and the Window Specification [page 424]

### 4.9.1.235 WIDTH_BUCKET Function (Miscellaneous)

Returns the bucket number that has been assigned to the result of a specified expression.

**Syntax**

```
WIDTH_BUCKET(<expression>, <min_value>, <max_value>, <num_buckets>)
```
Syntax Elements

- **expression**: Specifies the value that the histogram is created for. `<expression>` must be a numeric or datetime value, or a value that can be implicitly converted to a numeric or datetime value. If `<expression>` evaluates to NULL, then `<expression>` returns NULL.

- **min_value**: Specifies the minimum endpoint range for `<expression>`. `<min_value>` must be a numeric or datetime value and cannot evaluate to NULL.

- **max_value**: Specifies the maximum endpoint range for `<expression>`. `<max_value>` must be a numeric or datetime value and cannot evaluate to NULL.

- **num_buckets**: Specifies the number of buckets to distribute values across. `<num_buckets>` must be a positive integer.

Description

The returned bucket number is an integer between 0 and `<num_buckets>` +1. This function always returns `<num_buckets>` plus two additional buckets: bucket 0 and `<num_buckets>`+1. Bucket 0 contains the count of values less than `<min_value>` and bucket `<num_buckets>`+1 contains the count of values greater than or equal to `<max_value>`.

Example

The following example returns the bucket numbers for **myTable**:

```
CREATE TABLE myTable (myValues INT);
INSERT INTO myTable VALUES(1);
INSERT INTO myTable VALUES(2);
INSERT INTO myTable VALUES(3);
INSERT INTO myTable VALUES(4);
INSERT INTO myTable VALUES(5);
SELECT myValues, WIDTH_BUCKET( myValues, 2, 5, 2 ) AS "BucketNo" FROM myTable;
```

<table>
<thead>
<tr>
<th>myValues</th>
<th>BucketNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>75</td>
<td>3</td>
</tr>
</tbody>
</table>
Related Information

Expressions [page 56]

4.9.1.236 WORKDAYS_BETWEEN Function (Datetime)

Computes the number of workdays between a specified start date and a specified end date.

Syntax

```
WORKDAYS_BETWEEN(
    <factory_calendar_id>,
    <start_date>,
    <end_date>
    [, <source_schema>]
    [, <table_name_suffix>]
    [, <client>] ) )
```

Syntax Elements

- **factory_calendar_id**
  Specifies the ID of the factory calendar.

  ```
  <factory_calendar_id> ::= <string_literal>
  ```

- **start_date**
  Specifies the start date. You can use either a DAYDATE type or a date format string (for example '20140101' or '2014-01-01') for this parameter.

  ```
  <start_date> ::= <string_literal> | <DAYDATE>
  ```

- **end_date**
  Specifies the end date. You can use either a DAYDATE type or a date format string (for example '20140101' or '2014-01-01') for this parameter.

  ```
  <end_date> ::= <string_literal> | <DAYDATE>
  ```

- **source_schema**
  Specifies the schema containing the Factory Calendar table or the Factory And Holiday Calendar tables.

  ```
  <source_schema> ::= <string_literal>
  ```

  This parameter can be omitted if the schema is the same as the current schema.

- **table_name_suffix**
Specifies the suffix appended to the standard Factory Calendar table name or the standard Factory And Holiday Calendar table names if using an alternative set of tables. For example, with suffix '_XYZ' the function accesses the alternative tables TFACS_XYZ or FHC_C_FCAL_XYZ instead of standard tables TFACS or FHC_C_FCAL. If omitted or empty, the standard tables are used.

```sql
<table_name_suffix> ::= <string_literal>
```

client

Specifies the client used for the client-dependent Factory And Holiday Calendar implementation. If omitted or empty, then the function tries to retrieve the client from the context (user parameter), which can be set in the context by ALTER USER <user_name> SET PARAMETER CLIENT = <client>. The client is ignored if the client-independent Factory Calendar implementation is active.

```sql
<client> ::= <string_literal>
```

**Return Type**

INTEGER

**Description**

There are two different implementations for workday calculation in ABAP based systems like SAP S/4HANA:

- The Factory Calendar that stores its data in a single table TFACS.
- The Factory And Holiday Calendar that stores its data in multiple tables (FHC*).

By default, the WORKDAYS_BETWEEN functions assume that the Factory Calendar is active. Once table FHC_CONFIG (or FHC_CONFIG <table_suffix> where <table_suffix> is specified) is available in the system and contains an entry with KEY_FIELD = 'MIGRATION_COMPLETED' and VALUE = 'X' for the specified client, the WORKDAYS_BETWEEN functions assume that the newer Factory And Holiday Calendar is active.

The respective tables must be available in the SAP HANA database to use the WORKDAYS_BETWEEN function. In SAP BW, SAP CRM, and SAP ERP running on an SAP HANA database, these tables are located in the ABAP schema SAP<SID>. For other SAP HANA database systems, these tables can be replicated from an SAP Business Suite system.

When the Factory Calendar is active, WORKDAYS_BETWEEN accesses the table TFACS (or TFACS<table_suffix> if <table_suffix> is specified). When the Factory And Holiday Calendar is active, WORKDAYS_BETWEEN accesses the following tables, which must be available in the system with <table_suffix> appended to their respective names if <table_suffix> is specified:

- FHC_C_FCAL
- FHC_C_FCAL_EXC
- FHC_C_HCAL
- FHC_C_HCAL_ASSGN
- FHC_C_HOL
Computations are done with respect to a factory calendar with ID `<factory_calendar_id>`.

If the `<start_date>` is earlier than or equal to the `<end_date>`, then the number of working days in the period starting at the `<start_date>` and ending at the `<end_date>` is returned.

If the `<start_date>` is after the `<end_date>`, then a negative number of workdays in the period starting at the `<end_date>` and ending at the `<start_date>` is returned.

The returned number of days always includes the day of the `<end_date>`, but excludes the day of the `<start_date>`.

### Examples

The following examples use the Factory Calendar table for the month of January 2014 exclusively. For examples using the Factory And Holiday Calendar, see Factory Calendar.

<table>
<thead>
<tr>
<th>TFACS bitfield</th>
<th>Day of the month</th>
<th>Reason for not working</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>Public Holiday</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>Weekend</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>Weekend</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>Public Holiday</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>Weekend</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>Weekend</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>TFACS bitfield</td>
<td>Day of the month</td>
<td>Reason for not working</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>18</td>
<td>Weekend</td>
</tr>
<tr>
<td>0</td>
<td>19</td>
<td>Weekend</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>25</td>
<td>Weekend</td>
</tr>
<tr>
<td>0</td>
<td>26</td>
<td>Weekend</td>
</tr>
<tr>
<td>1</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

The following example returns the value 1 for work days between the two specified dates. **Result:** A single workday has been computed starting on the 9th and finishing on, but not including, the 10th.

```
SELECT WORKDAYS_BETWEEN('01', '2014-01-09', '2014-01-10' , 'FCTEST') "workdays"
FROM DUMMY;
```

The following example returns the value -1 for work days between the two specified dates. **Result:** A single workday has been computed on the 10th, excluding the day of the 9th.

```
SELECT WORKDAYS_BETWEEN('01', '2014-01-10', '2014-01-09' , 'FCTEST') "workdays"
FROM DUMMY;
```

The following example returns the value 1 for work days between the two specified dates. **Result:** The single workday has been computed on the 17th. The weekend period and the end date day are excluded.

```
SELECT WORKDAYS_BETWEEN('01', '2014-01-17', '2014-01-20' , 'FCTEST') "workdays"
FROM DUMMY;
```

The following example returns the value -1 for work days between the two specified dates. **Result:** The single workday has been computed on the 20th. The weekend period and the start date day are excluded.

```
SELECT WORKDAYS_BETWEEN('01', '2014-01-20', '2014-01-17' , 'FCTEST') "workdays"
FROM DUMMY;
```
The following example returns the value 20 for work days between the two specified dates. **Result:** The system takes into account the weekends (4th, 5th, 11th, 12th, 18th, 19th, 25th and 26th) and public holidays (1st and 6th) and excludes them from the working period.

```sql
SELECT WORKDAYS_BETWEEN('01', '2014-01-01', '2014-01-31', 'FCTEST') "workdays"
FROM DUMMY;
```

The following example returns the values 30, 28, 25, and 20.

```sql
SET SCHEMA "SAPABC";
CREATE ROW TABLE MY_DATES (FCID NVARCHAR(2), STARTDATE DAYDATE, ENDDATE DAYDATE);
INSERT INTO MY_DATES VALUES ('01', '2014-01-01', '2014-02-14');
INSERT INTO MY_DATES VALUES ('01', '2014-04-01', '2014-05-14');
INSERT INTO MY_DATES VALUES ('01', '2014-07-01', '2014-08-05');
INSERT INTO MY_DATES VALUES ('01', '2014-10-01', '2014-10-30');
SELECT WORKDAYS_BETWEEN(FCID, STARTDATE, ENDDATE) "production duration" FROM MY_DATES;
```

**Related Information**

- Data Types [page 33]
- Datetime Data Types [page 40]
- ALTER USER Statement (Access Control) [page 654]

### 4.9.1.237 XMLEXTRACT Function (Miscellaneous)

Returns an XML element that matches the specified XPath query.

**Syntax**

```sql
XMLEXTRACT(<XML_document>, <XPath_query> [,<NamespaceDeclarations>])
```

**Syntax Elements**

- **XML_document**
  Specifies an XML document of type CLOB, NCLOB, VARCHAR, or NVARCHAR.

- **XPath_query**
  Specifies an XPath expression of type VARCHAR or NVARCHAR.

- **NamespaceDeclarations**
  Specifies a namespace declaration of type VARCHAR or NVARCHAR.
Description
Returns the matching XML element. The return value is of type VARCHAR/NVARCHAR or CLOB/NCLOB
depending on the type given for <XML_document>.
If an XML element is empty (for example, <name></name>), then an empty result is returned. If an XML
element is not found, then the function returns an error.

Example
The following statement returns the <name> element from item 2 (the example returns <name>Jar</name>):
SELECT XMLEXTRACT(
'<doc>
<item><id>1</id><name>Box</name></item>
<item><id>2</id><name>Jar</name></item>
</doc>',
'/doc/item[2]/name'
) FROM DUMMY;
The following statement returns <ns1:name>Jar</ns1:name>:
SELECT XMLEXTRACT(
namespace2.sap.com">
<ns1:item><ns1:id>1</ns1:id><ns1:name>Box</ns1:name></ns1:item>
<ns1:item><ns1:id>2</ns1:id><ns1:name>Jar</ns1:name></ns1:item>
<ns2:item><ns2:id>3</ns2:id><ns2:name>Table</ns2:name></ns2:item>
</doc>',
'/doc/ns1:item[2]/ns1:name',
) FROM DUMMY;

Related Information
XMLEXTRACTVALUE Function (Miscellaneous) [page 405]

4.9.1.238 XMLEXTRACTVALUE Function (Miscellaneous)
Returns an XML value that matches the specified XPath query.

Syntax
XMLEXTRACTVALUE(<XML_document>, <XPath_query> [,<NamespaceDeclarations>])

SAP HANA SQL Reference Guide for SAP HANA Platform
SQL Reference

PUBLIC

405


Syntax Elements

XML_document
Specifies an XML document of type CLOB, NCLOB, VARCHAR, or NVARCHAR.

XPath_query
Specifies an XPath expression of type VARCHAR or NVARCHAR.

NamespaceDeclarations
Specifies a namespace declaration of type VARCHAR or NVARCHAR.

Description

Returns the text contents of the matching XML element. The return value is of type VARCHAR/NVARCHAR or CLOB/NCLOB depending on the type given for <XML_document>.

If an XML element is empty (for example, <name></name>), then an empty result is returned. If an XML element is not found, then the function returns an error.

This function is only supported for single elements of child XML nodes. For example, you can use XMLEXTRACTVALUE to query the <name> and <age> elements in the XML document below, as these are single elements of child nodes. However, you cannot query the <parent> or <child> elements of the document, as both of these elements contain multiple child nodes.

Example

The following example extracts the value from the <name> element from item 2 (and returns Jar):

```
SELECT XMLEXTRACTVALUE (    '<doc>'     '<item>'     '<id>1</id>'     '<name>Box</name>'     '</item>'    '<item>'     '<id>2</id>'     '<name>Jar</name>'     '</item>'    '</doc>' ) FROM DUMMY;
```

The following example extracts the value from the <name> element in item 2 (and returns Jar):

```
SELECT XMLEXTRACTVALUE (    '<doc xmlns:ns1="http://namespace1.sap.com" xmlns:ns2="http://namespace2.sap.com">'     '<ns1:item>'     '<ns1:id>1</ns1:id>'     '<ns1:name>Box</ns1:name>'     '</ns1:item>'     '<ns1:item>'     '<ns1:id>2</ns1:id>'     '<ns1:name>Jar</ns1:name>'     '</ns1:item>'     '<ns2:item>'     '<ns2:id>3</ns2:id>'     '<ns2:name>Table</ns2:name>'     '</ns2:item>'    '</doc>' ) FROM DUMMY;
```
4.9.1.239 XMLTABLE Function (String)

Creates a relational table from an XML string.

Syntax

```
XMLTABLE( [ <XML_namespace_clause>, ]   <row_pattern> PASSING <XML_argument> COLUMNS <column_definitions>   <error_option>  )
```

Syntax Elements

**XML_namespace_clause**

Defines the XML namespace.

```
XMLNAMESPACE (<XML_namespace>[, ...]   [DEFAULT <default_namespace>]  )
```

**XML_namespace**

Specifies the XML namespace.

```
<XML_namespace> ::= <namespace_url> AS <namespace_alias>
```

When the XMLNAMESPACE clause is specified, the result contains only the records that belong to the specified namespaces. If the namespace URL is specified (DEFAULT <namespace_url>), then all XPath strings should refer to the namespace URL.

**default_namespace**

Specifies the default namespace. <default_namespace>, <namespace_url>, and <namespace_alias> are string constants.

**row_pattern**
Describes the format of the xmlNodes in the XML with an XPath expression.

```plaintext
<row_pattern> ::= <str_const>
```

**XML_argument**

Specifies the XML to be converted to a table, defined as either as a string constant or a column reference.

```plaintext
<XML_argument> ::= <str_const> | <column_ref>
```

**str_const**

Specifies an XML string to be converted to a table.

**column_ref**

Specifies a column containing the XML to be converted to a table.

**column_definitions**

Defines the columns in the result table.

```plaintext
<column_definitions> ::= <column_name> <column_type> [, <column_name> <column_type>,...]
```

**column_name**

Specifies the column name.

```plaintext
<column_name> ::= <identifier>
```

**column_type**

Specifies a data type for the column.

```plaintext
<column_type> ::= FOR ORDINALITY | <regular_column_return_type> PATH <column_pattern> [DEFAULT <str_const>] | <formatted_column_return_type> FORMAT XML PATH <column_pattern> [DEFAULT <str_const>]
```

If the column is an ordinality column, then the value for the I-th row is I. The return type of the ordinality column is BIGINT.

If the column is a regular column, then the value for the I-th row can be defined as follows:

- If the result value is not empty, then the result comes from applying `<row_pattern>` or `<column_pattern>` to `<XML_argument>`.
- If the result value is empty, then the result comes from the default value of `<column_definition>`.
- If no default value is specified, then NULL is returned.
- If FORMAT XML is specified, then the result contains the XML tags and XML structure instead of returning the value.

**regular_column_return_type**

Specifies the return type of the column.

```plaintext
<regular_column_return_type> ::= 
```
column_pattern
Specifies the XPath Expression to use to extract the value for a column.

<column_pattern> ::= <str_const>

formatted_column_return_type
Specifies the list of possible return types for the formatted column in the resulting table.

<formatted_column_return_type> ::= VARCHAR (<int_const>) | NVARCHAR (<int_const>)

error_option
Returns an exception when an error occurs during table creation.

<error_option> ::= ERROR ON ERROR

If ERROR ON ERROR is not set, then an empty result is returned. If ERROR ON ERROR is specified, then a runtime error is returned instead of a DEFAULT or an empty result.

Description
Use the XMLTABLE function as a replacement for the FROM clause in a query to extract the data from an XML string. The XMLTABLE function returns the data as a relational table.

ERROR ON ERROR explicitly returns an error on a runtime error, since the default behavior is to return NULL in cases of a runtime error.

FORMAT XML extracts a result in XML format.

The resulting table has a row for each element in the XML string, which is the result of applying the <row_pattern> to the <XML_argument>. The rows of the resulting table have a column for each <column_definition>, with the <column_name> and <return_type> specified.

You cannot call the XMLTABLE function in the FROM clause of an UPDATE or DELETE statement.

Example
The following example returns the table below:

SELECT * FROM
The following example returns the table below:

```
SELECT * FROM
XMLTABLE('/doc/item' PASSING
'<!--doc--
  <item><id>10</id><name>Box</name></item>
  <item><id>20</id><name>Jar</name></item>
<!--doc-->
') COLUMNS
RN FOR ORDINAILITY,
NAME VARCHAR(20) PATH 'name'
) as XTABLE;
```

<table>
<thead>
<tr>
<th>RN</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Box</td>
</tr>
<tr>
<td>2</td>
<td>Jar</td>
</tr>
</tbody>
</table>

The following example returns the table below:

```
SELECT * FROM
XMLTABLE('/doc/item' PASSING
'<!--doc--
  <item><id>10</id><name>Box</name></item>
  <item><id>20</id><name>Jar</name></item>
<!--doc-->
') COLUMNS
ID NVARCHAR(20) FORMAT XML PATH 'id',
NAME VARCHAR(20) FORMAT XML PATH 'name'
) AS XTABLE;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;id&gt;10&lt;/id&gt;</td>
<td>&lt;name&gt;Box&lt;/name&gt;</td>
</tr>
<tr>
<td>&lt;id&gt;20&lt;/id&gt;</td>
<td>&lt;name&gt;Jar&lt;/name&gt;</td>
</tr>
</tbody>
</table>

The following example returns the table below:

```
SELECT * FROM
XMLTABLE('/doc/item' PASSING
'<!--doc--
  <item><id>10</id><name>Box</name></item>
  <item><id>20</id><name>Jar</name></item>
<!--doc-->
') COLUMNS
ID INT PATH 'id',
NAME VARCHAR(20) PATH 'name'
) as XTABLE;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Box</td>
</tr>
<tr>
<td>20</td>
<td>Jar</td>
</tr>
</tbody>
</table>
The following example returns the table below:

```sql
CREATE ROW TABLE XMLDATA (XML VARCHAR(5000));
INSERT INTO XMLDATA VALUES ('<sap:doc xmlns:sap="http://www.sap.com/xmltable">
  <sap:item><sap:id>1</sap:id><sap:name>Box</sap:name></sap:item>
  <sap:item><sap:id>2</sap:id><sap:name>Jar</sap:name></sap:item>
</sap:doc>');
INSERT INTO XMLDATA VALUES ('<labs:doc xmlns:labs="http://www.saplabskorea.com/xmltable">
  <labs:item><labs:id>1</labs:id><labs:name>Apple</labs:name></labs:item>
  <labs:item><labs:id>2</labs:id><labs:name>Coconut</labs:name></labs:item>
</labs:doc>');
SELECT * FROM XMLTABLE(XMLNAMESPACE('http://www.saplabskorea.com/xmltable' AS 'myns'),
'/myns:doc/myns:item' PASSING XMLDATA.XML
COLUMNS
  ID INT PATH 'myns:id',
  NAME VARCHAR(20) PATH 'myns:name'
) as XTABLE;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Box</td>
</tr>
<tr>
<td>2</td>
<td>Jar</td>
</tr>
<tr>
<td>ID</td>
<td>NAME</td>
</tr>
<tr>
<td>----</td>
<td>-------</td>
</tr>
<tr>
<td>1</td>
<td>Apple</td>
</tr>
<tr>
<td>2</td>
<td>Coconut</td>
</tr>
</tbody>
</table>

The following example returns the table below:

```sql
SELECT * FROM XMLTABLE(XMLNAMESPACE('http://www.sap.com/xmltable' AS 'myns', 'http://www.namespace.com/name' as 'myns2'),
'//myns:doc/myns:item' PASSING
'<?xml version="1.0" encoding="UTF-8"?>
  <item><id>1</id><name>Box</name><count>3</count></item>
  <item><id>2</id><name>Jar</name><count>4</count></item>
</doc>'
COLUMNS
  ID INT PATH 'myns:id',
  NAME VARCHAR(20) PATH 'myns:name',
  COUNT INT PATH 'myns2:count'
) AS XTABLE;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Box</td>
</tr>
<tr>
<td>2</td>
<td>Jar</td>
</tr>
</tbody>
</table>

The following example returns the table below:

```sql
SELECT * FROM XMLTABLE(XMLNAMESPACE('http://www.namespace.com/name' AS 'myns2' DEFAULT 'http://www.sap.com/xmltable'),
'/doc/item' PASSING
'<?xml version="1.0" encoding="UTF-8"?>
  <item><id>1</id><name>Box</name></item>
  <item><id>2</id><name>Jar</name></item>
</doc>'
COLUMNS
  ID INT PATH 'id',
  NAME VARCHAR(20) PATH 'myns2:name'
) AS XTABLE;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Box</td>
</tr>
<tr>
<td>2</td>
<td>Jar</td>
</tr>
</tbody>
</table>
4.9.1.240  YEAR Function (Datetime)

Returns the year number of a specified date.

Syntax

```
YEAR(<date>)
```

Description

Returns the year number of date `<date>`.

Example

The following example returns the value 2011 for the year of the specified date:

```
SELECT YEAR (TO_DATE ('2011-05-30', 'YYYY-MM-DD')) "year" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]

4.9.1.241  YEARS_BETWEEN Function (Datetime)

Computes the number of years between two specified dates.

Syntax

```
YEARS_BETWEEN(<date_1>, <date_2>)
```
Description

Computes the number of years between <date_1> and <date_2>. Returns NULL if either of <date_1> or <date_2> is NULL.

Examples

The following example returns the value 2 for the years between the two specified dates:

```
SELECT YEARS_BETWEEN(TO_DATE('2001-01-01'), TO_DATE('2003-03-14'))
"years_between" FROM DUMMY;
```

The following example returns the value -2 for the years between the two specified dates:

```
SELECT YEARS_BETWEEN(TO_DATE('2003-10-03'), TO_DATE('2001-01-14'))
"years_between" FROM DUMMY;
```

The following example returns the value 1 for the years between the two specified dates:

```
SELECT YEARS_BETWEEN(TO_DATE('2001-10-15'), TO_DATE('2003-01-14'))
"years_between" FROM DUMMY;
```

The following example returns the value 1 for the years between the two specified dates:

```
SELECT YEARS_BETWEEN(TO_DATE('2001-10-13'), TO_DATE('2003-01-14'))
"years_between" FROM DUMMY;
```

Related Information

Datetime Data Types [page 40]

4.9.2 Aggregate Functions

Aggregate functions are analytic functions that calculate an aggregate value based on a group of rows.

Supported functions:

- AUTO_CORR Function (Aggregate) [page 108]
- AVG Function (Aggregate) [page 110]
- CORR Function (Aggregate) [page 145]
Related Information

Window Aggregate Functions [page 428]

4.9.3 Array Functions

Array functions take arrays as input.

Supported functions:

• CARDINALITY Function (Array) [page 124]
• MEMBER_AT Function (Array) [page 264]
• SUBARRAY Function (Array) [page 353]
• TRIM_ARRAY Function (Array) [page 386]
4.9.4 Data Type Conversion Functions

Data type conversion functions convert data from one data type to another data type.

Data type conversion functions are used to convert function arguments from one data type to another, or to test whether a conversion is possible.

**i Note**

In both implicit and explicit numeric type conversions, these functions always truncate the least significant digits toward zero.

**Supported functions:**

- CAST Function (Data Type Conversion) [page 125]
- TO_ALPHANUM Function (Data Type Conversion) [page 364]
- TO_BIGINT Function (Data Type Conversion) [page 364]
- TO_BINARY Function (Data Type Conversion) [page 365]
- TO_BLOB Function (Data Type Conversion) [page 366]
- TO_BOOLEAN Function (Data Type Conversion) [page 367]
- TO_CLOB Function (Data Type Conversion) [page 368]
- TO_DATE Function (Data Type Conversion) [page 368]
- TO_DATS Function (Data Type Conversion) [page 369]
- TO_DECIMAL Function (Data Type Conversion) [page 370]
- TO_DOUBLE Function (Data Type Conversion) [page 371]
- TO_FIXEDCHAR Function (Data Type Conversion) [page 372]
- TO_INT Function (Data Type Conversion) [page 373]
- TO_INTEGER Function (Data Type Conversion) [page 373]
- TO_BOOLEAN (Data Type Conversion) [page 374]
- TO_NCHAR Function (Data Type Conversion) [page 375]
- TO_NVARCHAR Function (Data Type Conversion) [page 376]
- TO_REAL Function (Data Type Conversion) [page 377]
- TO_SECONDDATE Function (Data Type Conversion) [page 378]
- TO_SMALLDECIMAL Function (Data Type Conversion) [page 379]
- TO_SMALLINT Function (Data Type Conversion) [page 380]
- TO_TIME Function (Data Type Conversion) [page 380]
- TO_TIMESTAMP Function (Data Type Conversion) [page 381]
- TO_TINYINT Function (Data Type Conversion) [page 382]
- TO_VARCHAR Function (Data Type Conversion) [page 383]
4.9.5 Datetime Functions

Date and time functions perform operations on date and time data types or return date or time information.

Supported functions:

- ADD_DAYS Function (Datetime) [page 93]
- ADD_MONTHS Function (Datetime) [page 94]
- ADD_MONTHS_LAST Function (Datetime) [page 95]
- ADD_NANO100 Function (Datetime) [page 96]
- ADD_SECONDS Function (Datetime) [page 97]
- ADD_WORKDAYS Function (Datetime) [page 98]
- ADD_YEARS Function (Datetime) [page 102]
- CURRENT_DATE Function (Datetime) [page 167]
- CURRENT_MVCC_SNAPSHOT_TIMESTAMP Function (Datetime) [page 168]
- CURRENT_TIME Function (Datetime) [page 171]
- CURRENT_TIMESTAMP Function (Datetime) [page 172]
- CURRENT_UTCDATE Function (Datetime) [page 177]
- CURRENT.UTC_TIMESTAMP Function (Datetime) [page 178]
- CURRENT.UTC_TIMESTAMP Function (Datetime) [page 179]
- DAYNAME Function (Datetime) [page 180]
- DAYOFMONTH Function (Datetime) [page 180]
- DAYOFYEAR Function (Datetime) [page 181]
- DAYS_BETWEEN Function (Datetime) [page 182]
- EXTRACT Function (Datetime) [page 194]
- HOUR Function (Datetime) [page 212]
- ISOWEEK Function (Datetime) [page 219]
- LAST_DAY Function (Datetime) [page 237]
- LOCALTOUTC Function (Datetime) [page 249]
- MINUTE Function (Datetime) [page 268]
- MONTH Function (Datetime) [page 269]
- MONTHNAME Function (Datetime) [page 270]
- MONTHS_BETWEEN Function (Datetime) [page 271]
- NANO100_BETWEEN Function (Datetime) [page 272]
- NEXT_DAY Function (Datetime) [page 274]
- NOW Function (Datetime) [page 278]
- QUARTER Function (Datetime) [page 292]
- SECOND Function (Datetime) [page 319]
- SECONDS_BETWEEN Function (Datetime) [page 320]
- UTCTOLOCAL Function (Datetime) [page 390]
4.9.6 Fulltext Functions

Fulltext functions perform operations on data that has a fulltext index.

Supported functions:

- GROUP_SCORE Function (Miscellaneous) [page 199]
- INDEXING_ERROR_CODE Function (Fulltext) [page 214]
- INDEXING_ERROR_MESSAGE Function (Fulltext) [page 215]
- INDEXING_STATUS Function (Fulltext) [page 216]
- LANGUAGE Function (Fulltext) [page 236]
- MIMETYPE Function (Fulltext) [page 265]
- PLAINTEXT Function (Fulltext) [page 291]
- SCORE Function (Fulltext) [page 314]

4.9.7 Hierarchy Functions

Hierarchy functions allow you to work with hierarchical data such as tables with rows arranged in a tree or directed graph.

SAP HANA supports three types of hierarchy functions: generator, navigation, and scalar.

Following is a list of the supported hierarchy functions: generator, navigation, and scalar.

Generator functions

- HIERARCHY Generator Function
- HIERARCHY_LEVELED Generator Function
- HIERARCHY_SPANTREE Generator Function
- HIERARCHY_TEMPORAL Generator Function

Navigation functions

- HIERARCHY_ANCESTORS Navigation Function
- HIERARCHY_ANCESTORS_AGGREGATE Navigation Function
- HIERARCHY_DESCENDANTS Navigation Function
4.9.8 JSON Functions

JSON functions are functions that return or operate on JSON data.

Supported functions:

- **JSON_QUERY Function (JSON)** [page 220]
- **JSON_TABLE Function (JSON)** [page 223]
- **JSON_VALUE Function (JSON)** [page 231]
4.9.9 Miscellaneous Functions

SAP HANA supports many functions that return system values and perform various operations on values, expressions, and return values of other functions.

Supported functions:

- ALLOWPRECISION_LOSS Function (Miscellaneous) [page 103]
- COALESCE Function (Miscellaneous) [page 128]
- CONVERTCURRENCY Function (Miscellaneous) [page 131]
- CONVERSION_FUNCTION Function (Miscellaneous) [page 142]
- CURRENT_CONNECTION Function (Miscellaneous) [page 166]
- CURRENT_IDENTITY_VALUE Function (Miscellaneous) [page 168]
- CURRENT_OBJECT_SCHEMA Function (Miscellaneous) [page 169]
- CURRENT_SCHEMA Function (Miscellaneous) [page 170]
- CURRENT_SITE_ID Function (Miscellaneous) [page 170]
- CURRENT_TRANSACTION_ISO_ACTION_LEVEL Function (Miscellaneous) [page 173]
- CURRENT_UPDATE_STATEMENT_SEQUENCE Function (Miscellaneous) [page 174]
- CURRENT_UPDATE_TRANSACTION Function (Miscellaneous) [page 175]
- CURRENT_USER Function (Miscellaneous) [page 175]
- CURRENT_USER_ID Function (Miscellaneous) [page 176]
- EXPRESSION_MACRO Function (Miscellaneous) [page 193]
- GREATEST Function (Miscellaneous) [page 198]
- GROUPSCORE Function (Miscellaneous) [page 199]
- GROUPING Function (Miscellaneous) [page 202]
- GROUPING_ID Function (Miscellaneous) [page 204]
- HASH_MD5 Function (Miscellaneous) [page 208]
- HASH_SHA256 Function (Miscellaneous) [page 209]
- IFNULL Function (Miscellaneous) [page 213]
- LEAST Function (Miscellaneous) [page 242]
- MAP Function (Miscellaneous) [page 257]
- NEWSESSID Function (Miscellaneous) [page 275]
- NULLIF Function (Miscellaneous) [page 282]
- RECORD_COMMIT_TIMESTAMP Function (Miscellaneous) [page 299]
- RECORD_ID Function (Miscellaneous) [page 301]
- RESULT_CACHE_ID Function (Miscellaneous) [page 305]
- RESULT_CACHE_REFRESH_TIME Function (Miscellaneous) [page 306]
- SESSION_CONTEXT Function (Miscellaneous) [page 341]
- SESSION_USER Function (Miscellaneous) [page 342]
- SYSUUID Function (Miscellaneous) [page 362]
- WIDTH_BUCKET Function (Miscellaneous) [page 398]
4.9.10 Numeric Functions

Numeric functions perform mathematical operations on numerical data types or return numeric information.

Number functions take numeric values, or strings with numeric characters, as inputs and return numeric values. When strings with numeric characters are provided as inputs, implicit conversion from a string to a number is performed automatically before results are computed.

Supported functions:

- ABS Function (Numeric) [page 91]
- ACOS Function (Numeric) [page 92]
- ASIN Function (Numeric) [page 105]
- ATAN Function (Numeric) [page 106]
- ATAN2 Function (Numeric) [page 107]
- BITAND Function (Numeric) [page 117]
- BITCOUNT Function (Numeric) [page 118]
- BITNOT Function (Numeric) [page 119]
- BITOR Function (Numeric) [page 120]
- BITSET Function (Numeric) [page 121]
- BITUNSET Function (Numeric) [page 122]
- BITXOR Function (Numeric) [page 123]
- CEIL Function (Numeric) [page 126]
- COS Function (Numeric) [page 153]
- COSH Function (Numeric) [page 153]
- COT Function (Numeric) [page 154]
- EXP Function (Numeric) [page 192]
- FLOOR Function (Numeric) [page 196]
- LN Function (Numeric) [page 248]
- LOG Function (Numeric) [page 254]
- MOD Function (Numeric) [page 268]
- NDIV0 Function (Numeric) [page 273]
- POWER Function (Numeric) [page 292]
- RAND Function (Numeric) [page 293]
- RAND_SECURE Function (Numeric) [page 297]
- ROUND Function (Numeric) [page 308]
- SIGN Function (Numeric) [page 343]
4.9.11 Security Functions

Security functions provide special functionality for security purposes.

Supported functions:

- ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS Function (Security) [page 187]
- ENCRYPTION_ROOT_KEYS_HAS_BACKUP_PASSWORD Function (Security) [page 190]
- ESCAPE_DOUBLE_QUOTES Function (Security) [page 191]
- ESCAPE_SINGLE_QUOTES Function (Security) [page 191]
- GENERATE_PASSWORD Function (Security) [page 197]
- IS_SQL_INJECTION_SAFE Function (Security) [page 218]

4.9.12 Series Data Functions

Series data functions provide special functionality for series data and series tables.

Supported functions:

- SERIES_DISAGGREGATE Function (Series Data) [page 320]
- SERIES_ELEMENT_TO_PERIOD Function (Series Data) [page 325]
- SERIES_GENERATE Function (Series Data) [page 331]
- SERIES_PERIOD_TO_ELEMENT Function (Series Data) [page 335]
- SERIES_ROUND Function (Series Data) [page 337]
4.9.13 String Functions

String functions perform extraction and manipulation on strings, or return information about strings.

Supported functions:

- ABAP_ALPHANUM Function (String) [page 84]
- ABAP_DF16RAW_TO_SMALLDECIMAL Function (String) [page 86]
- ABAP_DF34RAW_TO_DECIMAL Function (String) [page 87]
- ABAP_LOWER Function (String) [page 89]
- ABAP_NUMC Function (String) [page 88]
- ABAP_UPPER Function (String) [page 90]
- ASCII Function (String) [page 104]
- BINTOHEX Function (String) [page 114]
- BINTONHEX Function (String) [page 115]
- BINTOSTR Function (String) [page 116]
- CHAR Function (String) [page 127]
- CONCAT Function (String) [page 129]
- CONCAT_NAZ Function (String) [page 130]
- HAMMING_DISTANCE Function (String) [page 206]
- HEXTOBIN Function (String) [page 210]
- HEXTONUM Function (String) [page 211]
- INITCAP Function (String) [page 217]
- LCASE Function (String) [page 239]
- LEFT Function (String) [page 242]
- LENGTH Function (String) [page 243]
- LOCATE Function (String) [page 250]
- LOCATE_REGEXPR Function (String) [page 252]
- LOWER Function (String) [page 255]
- LPAD Function (String) [page 255]
- LTRIM Function (String) [page 257]
- NCHAR Function (String) [page 273]
- NORMALIZE Function (String) [page 276]
- NUMTOHEX Function (String) [page 283]
- OCCURRENCES_REGEXPR Function (String) [page 284]
- REPLACE Function (String) [page 302]
- REPLACE_REGEXPR Function (String) [page 303]
- RIGHT Function (String) [page 307]
- RPAD Function (String) [page 310]
- RTRIM Function (String) [page 311]
4.9.14 Window Functions and the Window Specification

Window functions allow you to perform analytic operations over a set of input rows.

Syntax

<function_name> <window_specification>

>window_specification> ::= OVER ( { [ <window_partition_by_clause> ] [ [ <window_order_by_clause> ] [ <window_frame_clause> ] ] } )

Syntax Elements

function_name

Specifies the window function to operate over the data. See the individual function topics for more information on their syntax.

<function_name> ::= BINNING
     | CUBIC_SPLINE_APPROX
     | CUME_DIST
     | DENSE_RANK
     | LAG
     | LEAD
     | LINEAR_APPROX
     | NTILE
     | PERCENT_RANK
     | PERCENTILE_CONT
     | PERCENTILE_DISC
     | RANDOM_PARTITION
     | RANK
     | ROW_NUMBER
     | SERIES_FILTER
For `<window_aggregate_functions>`, see Window Aggregate Functions [page 428].

For `<spatial_functions>`, see the SAP HANA Spatial Reference guide.

**window_partition_by_clause**

<window_partition_by_clause> ::= PARTITION BY <expression> [ { , <expression> } ... ]

Creates one or more window partitions using the supplied expressions.

**window_order_by_clause**

Defines the sort order of the window.

>window_order_by_expression> ::= ORDER BY <window_order_by_expression> [ {, <window_order_by_expression> } ... ]
<window_order_by_expression> ::= <column> [ [ASC | DESC] [ NULLS { FIRST | LAST } ] ]
<collate_clause> ::= COLLATE <collation_name>

**collate_clause**

Specifies the collation to use. `<collate_clause>` can only be used on columns defined as NVARCHAR or VARCHAR.

<collation_name> is one of the supported collation names listed in the COLLATIONS system view.

[ASC | DESC]

ASC sorts records in ascending order and DESC sorts records in descending order. The default is ASC.

[NULLS FIRST | NULLS LAST]

Specifies the ordering of NULL values.

**position**

<position> uses the entries in the select list to define the ordering required. For example:

SELECT col1, col2 FROM t ORDER BY 2;

ORDER BY 2 indicates that ordering should be undertaken using the second expression in the select list, which in this case is col2.

**window_frame_clause**

Defines the rows (bounds) of a window partition by specifying the number of rows preceding or following with respect to the current row. If you specify a `<window_frame_clause>`, you must also specify a `<window_order_by_clause>`.

>window_frame_clause> ::= <window_frame_unit> <window_frame_extent>
>window_frame_extent> ::= { { <window_frame_start> | <window_frame_between> } }
>window_frame_unit> ::= ROWS
>window_frame_start> ::= { UNBOUNDED PRECEDING | <window_frame_preceding> | CURRENT ROW }
>window_frame_preceding> ::= { <unsigned_integer> PRECEDING
>window_frame_between> ::= BETWEEN <lower_window_frame_bound> AND
<lower_window_frame_bound> ::= <window_frame_bound>
<upper_window_frame_bound> ::= <window_frame_bound>
\(<window\_frame\_bound> := \{ <window\_frame\_start> | UNBOUNDED \text{FOLLOWING} | \}
<window\_frame\_between> ::= <unsigned\_integer> \text{FOLLOWING}

\(<window\_frame\_followings> := <unsigned\_integer> \text{FOLLOWING}

\(<\text{window\_frame\_between}> \text{specifies the lower (starting) bound and the upper (ending) bound of the window frame. The upper bound cannot be smaller than the lower bound.}

\text{UNBOUND PRECEDING specifies that the window starts at the first row of the partition. It can only be specified as <window\_frame\_start>. CURRENT ROW specifies that the window starts or ends at the current row when used with ROWS.}

\text{PRECEDING indicates that the number of rows or values to precede the current row.}

\text{<unsigned\_integer> should be 0 or a positive integer literal.}

\text{UNBOUNDED FOLLOWING specifies that the window ends at the last row of the partition. UNBOUNDED FOLLOWING can only be specified as a window frame end.}

\text{FOLLOWING indicates the number of rows or values to follow the current row. When specified as the window frame start, the window frame end should be <unsigned\_integer> \text{FOLLOWING}. The <unsigned\_integer> should be 0 or a positive integer literal.}

\text{Description}

\text{The window functions allow you to divide the result sets of a query, or a logical partition of a query, into groups of rows called window partitions.}

\text{A window partition is specified by one or more expressions in the OVER clause.}

\text{Window functions not in the following list must have an ORDER BY clause in the OVER clause:}

\text{• AVG}
\text{• CORR}
\text{• CORR\_SPEARMAN}
\text{• COUNT}
\text{• MAX}
\text{• MEDIAN}
\text{• MIN}
\text{• PERCENTILE\_CONT}
\text{• PERCENTILE\_DISC}
\text{• ROW\_NUMBER}
\text{• STDDEV}
\text{• SUM}
\text{• VAR}

\text{Result sets are first partitioned as specified by PARTITION BY clause, and then sorted by ORDER BY clause specification within the window partition.}

\text{Finally window functions are applied to each row within window partition boundaries.}

\text{The ORDER BY clause in the OVER clause is only used to evaluate window function so that the order of resulting rows is non-deterministic if not specified by ORDER BY for SELECT.}
The default window frame of the window function depends on whether or not a window ORDER BY clause is specified. If a window ORDER BY clause is specified, the default window frame becomes 'between UNBOUNDED PRECEDING and CURRENT ROW'. That is to say that the window function computes on rows preceding or peer with current row. As a result, the function returns cumulative values.

If a window ORDER BY clause is not specified, the default window frame becomes 'between UNBOUNDED PRECEDING and UNBOUNDED FOLLOWING'. That is to say that the window function computes on the whole window partition. As a result, the function returns the same value within a window partition regardless of current row.

**Examples**

The examples use the example table T defined below.

```
CREATE TABLE myTable (Class CHAR(10), Value INT, Offset INT);
INSERT INTO myTable VALUES('A', 1, 1);
INSERT INTO myTable VALUES('A', 3, 3);
INSERT INTO myTable VALUES('A', 5, NULL);
INSERT INTO myTable VALUES('A', 5, 2);
INSERT INTO myTable VALUES('A', 10, 0);
INSERT INTO myTable VALUES('B', 1, 3);
INSERT INTO myTable VALUES('B', 1, 1);
INSERT INTO myTable VALUES('B', 7, 1);
```

```
SELECT Class, Value, ROW_NUMBER() OVER (PARTITION BY Class ORDER BY Value) AS row_num,
       RANK() OVER (PARTITION BY Class ORDER BY Value) AS rank,
       DENSE_RANK() OVER (PARTITION BY Class ORDER BY Value) AS dense_rank
FROM myTable;
```

<table>
<thead>
<tr>
<th>class</th>
<th>val</th>
<th>row_num</th>
<th>rank</th>
<th>dense_rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
Related Information

Window Aggregate Functions [page 428]
Spatial Functions
BINNING Function (Window) [page 112]
CUBIC_SPLINE_APPROX Function (Window) [page 159]
COLLATIONS System View [page 1520]
CUME_DIST Function (Window) [page 164]
DENSE_RANK Function (Window) [page 183]
LAG Function (Window) [page 234]
LEAD Function (Window) [page 240]
LINEAR_APPROX Function (Window) [page 244]
NTILE Function (Window) [page 281]
PERCENT_RANK Function (Window) [page 286]
PERCENTILE_CONT Function (Window) [page 287]
PERCENTILE_DISC Function (Window) [page 289]
RANDOM_PARTITION Function (Window) [page 294]
RANK Function (Window) [page 298]
ROW_NUMBER Function (Window) [page 312]
SERIES_FILTER Function (Window) [page 327]
WEIGHTED_AVG Function (Window) [page 396]
SAP HANA Spatial Reference

4.9.14.1 Window Aggregate Functions

Some aggregate functions can be used as window functions over a window specification.

Syntax

```sql
<aggregate_function_or_expression_as_window_function> ::= 
<aggregate_function_name> ( <arguments> ) <window_specification> 
<aggregate_function_name> ::= 
AVG
CORR
CORR_SPEARMAN
COUNT
FIRST_VALUE
LAST_VALUE
MAX
MEDIAN
MIN
NTH_VALUE
STDDEV
SUM
VAR
```
Syntax Elements

**aggregate_function_name**
Specifies the aggregate function or aggregate expression to use as a window function.

**arguments**
Specifies the input to the function. Refer to the individual function topic for the definition of `<arguments>`.

**window_specification**
Specifies the window definition. See Window Functions and the Window Specification [page 424]

Description

Window aggregate functions are aggregate functions that can be used as window functions.

Examples

```
CREATE ROW TABLE T (class CHAR(10), val INT, offset INT);
INSERT INTO T VALUES('A', 1, 1);
INSERT INTO T VALUES('A', 3, 3);
INSERT INTO T VALUES('A', 5, null);
INSERT INTO T VALUES('A', 5, 2);
INSERT INTO T VALUES('A', 10, 0);
INSERT INTO T VALUES('B', 1, 3);
INSERT INTO T VALUES('B', 1, 1);
INSERT INTO T VALUES('B', 7, 1);
SELECT class, val, offset,
 COUNT(*) OVER (PARTITION BY class) AS c1,
 COUNT(offset) OVER (PARTITION BY class) AS c2,
 COUNT(*) OVER (PARTITION BY class ORDER BY val) AS c3,
 COUNT(offset) OVER (PARTITION BY class ORDER BY val) AS c4,
 MAX(val) OVER (PARTITION BY class) AS m1,
 MAX(val) OVER (PARTITION BY class ORDER BY val) AS m2
 FROM T;
```

<table>
<thead>
<tr>
<th>CLASS</th>
<th>VAL</th>
<th>OFFSET</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>null</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>A</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
4.10 SQL Statements

SAP HANA supports many SQL statements to allow you to perform such tasks as create database objects, administer your system, and manipulate data.

- Alphabetical List of Statements [page 431]
- Access Control Statements [page 1199]
  Access control statements enable database administrators to create, alter, and drop access to the SAP HANA database.
- Backup and Recovery Statements [page 1200]
  These statements allow users to perform backup and recovery operations.
- Data Definition Statements [page 1201]
  Data definition statements define structures in the SAP HANA database.
- Data Import Export Statements [page 1202]
  The following statements import and export data to and from the SAP HANA database.
Data Manipulation Statements [page 1203]
The following statements enable you to manipulate data within the SAP HANA database.

Client-side Encryption Statements [page 1203]
Client-side encryption statements enable database administrators to administer client-side encrypted data in the SAP HANA database.

JSON Document Store Statements [page 1204]
There are several JSON Document Store-specific variants of common DDL and DML SQL statements.

Procedural Statements [page 1204]
Procedural statements manage system and user-defined procedures for the SAP HANA database.

Session Management Statements [page 1205]
The following SQL statements manage database sessions.

System Management Statements [page 1205]
The following statements enable you to manage system configuration settings in your SAP HANA database.

Tenant Database Management Statements [page 1207]
The following SQL statements manage SAP HANA tenant databases.

Transaction Management Statements [page 1208]
The following SQL statements manage transactions in the SAP HANA database.

Workload Management Statements [page 1208]
The following SQL statements manage workload classes and mappings.

4.10.1 Alphabetical List of Statements

4.10.1.1 ALTER AUDIT POLICY Statement (Access Control)

Enables or disables an audit policy, changes audit trail target types for an audit policy, and configures the retention period of the policy.

Syntax

```
ALTER AUDIT POLICY <policy_name>   { [ FOR <database_name> ] <audit_mode>   
   <set_audit_trail_type>      
   <reset_audit_trail_type> 
   <set_audit_trail_retention> 
   <reset_audit_trail_retention> }
```
**Syntax Elements**

`policy_name`
Specifies the name of the audit policy to be altered.

```plaintext
<policy_name> ::= <identifier>
```

`database_name` Defines the name of the tenant the audit policy is applied to. This option can only be used on SYSTEMDB, not on a tenant in conjunction with the `audit_mode` option.

**audit_mode**
Enables or disables the audit policy.

```plaintext
<audit_mode> ::= ENABLE | DISABLE
```

**ENABLE**
Enables the audit policy.

**DISABLE**
Disables the audit policy.

`set_audit_trail_type`
Specifies the audit trail target types for the audit policy.

```plaintext
<set_audit_trail_types> ::= SET TRAIL TYPE <audit_trail_type_list>
<audit_trail_type_list> ::= <audit_trail_type_name> [, ]
<audit_trail_type_name> ::= TABLE [ RETENTION <duration> ] | SYSLOG | CSV
```

Specifying CSV on a production system has security implications: users with the DATA ADMIN, CATALOG READ, TRACE ADMIN, or INIFILE ADMIN privilege are able to access the CSV file.

**duration** Specifies the length of time the policy is retained, expressed in days. The value specified in the CREATE or ALTER AUDIT POLICY statement must be greater than the value specified for the minimal_retention_period parameter in the auditing configuration section of the global.ini file. An error is generated if the value is less. The minimal_retention_period parameter value must be between 7 and 3653 (10 years). This limitation does not apply to the duration value in the CREATE or ALTER AUDIT POLICY statement.

`reset_audit_trail_type`
Resets the audit trail target type(s) to the system default.

```plaintext
<reset_audit_trail_type> ::= RESET TRAIL TYPE
```

`set_audit_trail_retention`
Specifies the length of time the audit entries are retained.

```plaintext
<set_audit_trail_retention> ::= SET RETENTION <duration>
```

`reset_audit_trail_retention`
Resets the audit trail retention period to the default, which is forever.

\texttt{\textbackslash reset\_audit\_trail\_retention} \texttt{::= \text{RESET \text{RETENTION}}}

\textbf{i Note}

If the retention period is changed, any audit entries that are no longer included are deleted immediately.

\textbf{Description}

The \texttt{ALTER AUDIT POLICY} statement enables or disables an audit policy. \texttt{\langle policy\_name\rangle} must specify an existing audit policy. Only database users with the \texttt{AUDIT ADMIN} privilege are allowed to alter an audit policy. Users with this privilege can alter any audit policy, regardless of if they are the creator of the policy.

New audit policies are disabled by default and must be enabled before the audit actions begin. The configuration parameter \texttt{global\_auditing\_state} must also be set to \texttt{true}.

An audit policy can be disabled and enabled as often as required.

Specify one or more audit trail targets for an audit policy at the time of creation. The allowed audit trail targets are:

- \texttt{SYSLOG}: uses the system syslog.
- \texttt{TABLE}: stores audit information in database table. The audit log is accessible using \texttt{AUDIT\_LOG} system view.
- \texttt{CSV}: stores audit information as comma-separated values in a text file. Use only for testing purposes.

\textbf{Permissions}

To alter an audit policy on \texttt{SYSTEMDB} or on a tenant, you must have the \texttt{AUDIT ADMIN} privilege.

To use the \texttt{FOR \langle database\_name\rangle} option, you need the \texttt{DATABASE AUDIT ADMIN} privilege granted on \texttt{SYSTEMDB}, not on the tenant.

\textbf{Example}

Create an audit policy called \texttt{priv\_audit} by using the following statement.

\texttt{CREATE AUDIT POLICY "priv\_audit" AUDITING SUCCESSFUL GRANT PRIVILEGE, REVOKE PRIVILEGE, GRANT ROLE, REVOKE ROLE LEVEL CRITICAL;}

Enable the \texttt{priv\_audit} audit policy.

\texttt{ALTER AUDIT POLICY "priv\_audit" ENABLE;}
Set SYSLOG and TABLE as audit trail targets for the priv_audit audit policy.

```sql
ALTER AUDIT POLICY "priv_audit" SET TRAIL TYPE SYSLOG, TABLE;
```

Reset audit trail targets for the priv_audit audit policy.

```sql
ALTER AUDIT POLICY "priv_audit" RESET TRAIL TYPE;
```

Disable the priv_audit audit policy.

```sql
ALTER AUDIT POLICY "priv_audit" DISABLE;
```

Set the retention period for the priv_audit policy to 30 days.

```sql
ALTER AUDIT POLICY "priv_audit" SET RETENTION 30;
```

Remove the retention period for the priv_audit policy.

```sql
ALTER AUDIT POLICY "priv_audit" RESET RETENTION;
```

Enable the audit policy SYSTEMDB_TEST, which was created on the SYSTEMDB and applies the tenant HA2 only.

```sql
ALTER AUDIT POLICY SYSTEMDB_TEST FOR HA2 ENABLE;
```

**Related Information**

- [AUDIT_POLICIES System View](#)
- [CREATE AUDIT POLICY Statement (Access Control)](#)
- [System Properties for Configuring Auditing](#)
- [Auditing Activity in SAP HANA](#)
- [CREATE AUDIT POLICY Statement (Access Control)](#)
- [AUDIT_POLICIES System View](#)
- [M_INIFILE_CONTENTS System View](#)
- [AUDIT_LOG System View](#)
4.10.1.2 ALTER CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption)

ALTER CLIENTSIDE ENCRYPTION COLUMN KEY [ <schema_name>.]<key_name>    [ { ADD | DROP } KEYCOPY ] ENCRYPTED WITH KEYPAIR <keypair_name>
| ADD NEW VERSION
| DROP OLD VERSIONS

Syntax Elements

schema_name

Specifies the schema name for the CEK.

<schema_name> ::= <identifier>

key_name

Specifies the name of the CEK to alter.

{ ADD | DROP } KEYCOPY

Specifies to create (ADD) or drop a key copy of the CEK.

ENCRYPTED WITH KEYPAIR keypair_name

Specifies the name of the client key pair (CKP) to use for encrypting the column key.

<keypair_name> ::= <identifier>

ADD NEW VERSION Creates a new version of the encryption key.

DROP OLD VERSIONS Removes all old versions of the encryption key. Encrypted columns that are using the old version(s) of the CEK must be re-encrypted using the new version of the CEK before the old version can be dropped.

Description

When a column encryption key is created by the key administrator (user with CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege), no other user will have access to this CEK and there is only one encrypted value for the CEK. When a particular client installation requires access to a CEK, the client creates a client key pair using the CREATE CLIENTSIDE ENCRYPTION KEYPAIR<keypair_name> syntax. When the key administrator uses the ADD KEYCOPY clause, the client driver collaborates with the SAP HANA server to
create a copy of the CEK that is encrypted with the key pair name specified in the statement (ENCRYPTED WITH KEYPAIR <keypair_name>) and created previously by the client. The key value of the CEK copy is added to the SAP HANA database, thus granting access to the CEK and the data encrypted with this CEK to the client installation.

With DROP KEYCOPY, access to the CEK is removed from the SAP HANA database. If the key copy being dropped belongs to the key administrator, then the SAP HANA server ensures that at least one other key administrator key copy (CEK copy encrypted by a CKP owned by the key administrator) exists for that CEK. If no other key administrator key copies exist for that CEK, then DROP KEYCOPY fails. This behavior ensures that the key administrator can create key copies for the CEK that grants access to encrypted data.

When an empty CEK is created using a CREATE CLIENTSIDE ENCRYPTION COLUMN KEY...HEADER ONLY statement, there is no encrypted value for the CEK in the SAP HANA database. An empty CEK allows the client to proceed with database schema creation without actually creating the CEK. With the ENCRYPTED WITH KEYPAIR <keypair_name> syntax, the client driver collaborates with the SAP HANA server to generate a new CEK and encrypts it with <keypair_name>, which is then stored in the SAP HANA database. Once the empty CEK is populated, rows can be inserted into the table. The {ADD | DROP} KEYCOPY clause is not used to populate the empty CEK.

The ADD VERSION clause creates a new version of an existing CEK. The ALTER TABLE <activate_clientside_encryption> clause then applies the new CEK version to the encrypted column.

Permissions

This statement requires the CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege.

Examples

The following statement creates a key copy of the CEK hrapp_cek1 for user key pair user1_ckp. This statement allows user1 to access CEK hrapp_cek1 and thus the encrypted column that the CEK encrypts.

```
ALTER CLIENTSIDE ENCRYPTION COLUMN KEY hrapp_cek1 ADD KEYCOPY ENCRYPTED WITH KEYPAIR user1_ckp;
```

The following statement drops the key copy of CEK hrapp_cek1 encrypted with key pair user1_ckp. This statement prevents the client from having access to hrapp_cek1 through the client key pair user1_ckp.

```
ALTER CLIENTSIDE ENCRYPTION COLUMN KEY hrapp_cek1 DROP KEYCOPY ENCRYPTED WITH KEYPAIR user1_ckp;
```

The following statement alters an empty CEK hrapp_cek2 (defined as HEADER ONLY) and encrypts with user1_ckp.

```
CREATE CLIENTSIDE ENCRYPTION COLUMN KEY hrapp_cek2 HEADER ONLY;
ALTER CLIENTSIDE ENCRYPTION COLUMN KEY hrapp_cek2 ENCRYPTED WITH KEYPAIR user1_ckp;
```

The following statement adds a new version of the existing CEK hrapp_cek3.

```
ALTER CLIENTSIDE ENCRYPTION COLUMN KEY hrapp_cek2 ADD NEW VERSION;
```
The following statement drops an old version of the existing CEK hrapp_cek3.

```
ALTER CLIENTSIDE ENCRYPTION COLUMN KEY hrapp_cek2 DROP OLD VERSIONS;
```

**Related Information**

- CLIENTSIDE_ENCRYPTION_COLUMN_KEYS System View [page 1517]
- CLIENTSIDE_ENCRYPTION_KEYPAIRS System View [page 1519]
- CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 741]
- DROP CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 949]
- CREATE CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 743]
- DROP CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 950]
- ALTER TABLE Statement (Data Definition) [page 589]

**4.10.1.3 ALTER CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption)**

Adds a new version of a client-side encryption key pair (CKP), or drops all older versions of a CKP.

**Syntax**

```
ALTER CLIENTSIDE ENCRYPTION KEYPAIR <keypair_name> <add_drop_version>
```

**Syntax Elements**

- **keypair_name**
  Specifies the name of an existing CKP.
- **add_drop_versions**
  Adds a new version of the specified CKP, or drops all previous versions.

```
<add_drop_versions> ::= { ADD NEW VERSION | DROP OLD VERSIONS }
```
**Description**

The client-driver uses this statement to create a new version of an existing CKP which the SAP HANA server then stores in the catalog, or to delete all older versions of a CKP.

Dropping of old versions of CKP also drops the key copies of the CEK that they encrypt.

Dropping old versions fails if any of the older versions of CKP encrypts the last key copy of a CEK version.

If all key copies of CEK encrypted by older versions of CKP are encrypted with latest version of CKP which belongs to the key admin, then older versions of CKP are dropped along with the key copies of CEK that they encrypt.

**Permissions**

To add a new version, you must be the owner of the CKP and have ALTER CLIENTSIDEN ENCRYPTION KEYPAIR privilege. To drop old versions, you must have the ALTER CLIENTSIDEN ENCRYPTION KEYPAIR privilege.

**Examples**

The following example creates a CKP, **Ckp1**, creates a new version of it, and then drops all old versions of it.

```
CREATE CLIENTSIDE ENCRYPTION KEYPAIR Ckp1 ALGORITHM 'RSA-OAEP-2048';
ALTER CLIENTSIDE ENCRYPTION KEYPAIR Ckp1 ADD NEW VERSION;
ALTER CLIENTSIDE ENCRYPTION KEYPAIR Ckp1 DROP OLD VERSIONS;
```

**Related Information**

- CREATE CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 743]
- CLIENTSIDE_ENCRYPTION_COLUMN_KEYS System View [page 1517]
- CLIENTSIDE_ENCRYPTION_KEYPAIRS System View [page 1519]
- CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 741]
- ALTER CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 435]
- DROP CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 949]
- DROP CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 950]
- hdbkeystore Utility
4.10.1.4 ALTER CREDENTIAL Statement (Access Control)

Modifies an existing component-specific or application-specific credential.

Syntax

```
ALTER CREDENTIAL FOR [USER <user_name>] COMPONENT <component_id>     PURPOSE <purpose_def> TYPE <type_def> [USING <using_param>]
```

Syntax Elements

- **user_name**
  Specifies the user that owns the credential.
  
  `<user_name> ::= <unicode_name>`
  
  Only users with the CREDENTIAL ADMIN system privilege are allowed to alter credentials for other users.

- **component_id**
  Specifies an identifier for the component that uses the credential.
  
  `<component_id> ::= <string_literal>`

- **purpose_def**
  Specifies the application-specific or component-specific purpose string.
  
  `<purpose_def> ::= <string_literal>`

- **type_def**
  Specifies a connection mechanism, such as ’PASSWORD’ or ’KERBEROS’.
  
  `<type_def> ::= <string_literal>`

- **using_param**
  Specifies the connection parameter. This is using a type specified by the `<type_def>` parameter. This value is required if the type is ’PASSWORD’.
  
  `<using_param> ::= <string_literal>`

Description

Only the credentials can be changed using this command. Each user can alter their own credentials.
Permissions

This statement requires the CREDENTIAL ADMIN privilege.

Example

Create a credential for all users that is used by the component INTERNAL_APP. This credential uses the password mechanism for the purpose COMPANY_MASTER_MACHINE.

```
CREATE CREDENTIAL FOR COMPONENT 'INTERNAL_APP' PURPOSE 'COMPANY_MASTER_MACHINE'
  TYPE 'PASSWORD' USING 'PASSWORD_9876';
```

Change the password of the credential created in the previous step.

```
ALTER CREDENTIAL FOR COMPONENT 'INTERNAL_APP' PURPOSE 'COMPANY_MASTER_MACHINE'
  TYPE 'PASSWORD' USING '1234_PASSWORD';
```

Related Information

CREDENTIALS System View [page 1524]

4.10.1.5 ALTER DATABASE Statement (Tenant Database Management)

Makes changes to a tenant database.

Syntax

```
ALTER DATABASE <database_name>
  <alter_service_spec_list>
  | ALTER <service_type> AT [ LOCATION ] '<hostname>[:<port_number>]' TO
  | LOCATION | '<new_port_number>'
  | OS USER '<username>' | OS GROUP '<groupname>'
  | FINALIZE REPLICA [ DROP SOURCE DATABASE ]
  | CANCEL REPLICA
  | SYSTEM USER PASSWORD <password>
  | { NO | DEFAULT } RESTART
  | ENCRYPTION CONFIGURATION CONTROLLED BY LOCAL DATABASE
  | { PERSISTENCE | LOG | BACKUP } ENCRYPTION [ ON | OFF ]
  | { CREATE | DROP } FALLBACK SNAPSHOT
  | <key_management_spec_list>
  | UNSET LICENSE ALL
```
### Syntax Elements

**database_name**

Specifies the name of the tenant database to which the change is applied.

\[
\text{<database\_name>} ::= \text{<identifier>}
\]

**alter_service_spec_list**

Adds or removes a service, or alters the port of a service.

\[
\text{<alter\_service\_spec\_list>} ::= \text{<alter\_service\_spec>} \ [\text{<alter\_service\_spec>}…]\]  
\text{<alter\_service\_spec>} ::= \text{ADD}'<\text{service\_type}>' \ [\text{AT} \ [\text{LOCATION}] \ '<\text{hostname}>'][\text{:<port\_number}>]' \ ]  
\ | \text{REMOVE}'<\text{service\_type}>' \ [\text{AT} \ [\text{LOCATION}] \ '<\text{hostname}>'][\text{:<port\_number}>]'\]
\]

**service_type**

Specifies the service type to add.

\[
\text{<service\_type>} ::= \text{indexserver} | \text{xsengine} | \text{scriptserver} | \text{dpserver} | \text{diserver} | \text{docstore}
\]

See the *SAP HANA Administration Guide* for more information about services.

**hostname**

Specifies the host name where the tenant database is located.

**port_number**

Specifies the port number for the host name where the tenant database is located.

**new_port_number**

Specifies a different port number for the host name where the tenant database is located.

**username**

Changes the user name.

\[
\text{<username>} ::= \text{<identifier>}
\]

**groupname**

Changes the OS user and OS group.

\[
\text{<groupname>} ::= \text{<identifier>}
\]

The tenant database must be offline to change the user and group. The user and group must already exist. If you specify *OS USER '' OS GROUP ''*, then it is interpreted as *OS USER 'sidadm' OS GROUP 'sapsys'*.  

**FINALIZE REPLICA**

Stops replication and finalizes the copy of the tenant database. Use this syntax when the replication status is ACTIVE.

**DROP SOURCE DATABASE**

Drops the original tenant database in the source system.

**CANCEL REPLICA**
Cleans up the system. Use this syntax when you cannot create a new replica because the replication status of the source database is REPLICATING even though the target database has been dropped. Execute this statement from the source database (the one that is being replicated).

**SYSTEM USER PASSWORD password**

Specifies a new system user password for the tenant database.

**{NO | DEFAULT} RESTART**

Specifies the restart mode for the tenant database after a system restart.

DEFAULT restores the previous state of the tenant database before the system restart. For example, if the tenant database was not started before the system restart, it is not started after the system restart. This is the default behavior.

NO specifies to not restart the tenant database after a system restart.

**ENCRYPTION CONFIGURATION CONTROLLED BY LOCAL DATABASE**

Specifies that encryption is controlled by the specified tenant database.

**{ PERSISTENCE | LOG | BACKUP } ENCRYPTION [ ON | OFF ]**

Turns encryption on or off for the specified tenant database. This command only works if the system database controls the encryption configuration of the tenant.

**{ CREATE | DROP } FALLBACK SNAPSHOT**

Creates or removes a fallback snapshot for the tenant database, respectively. A fallback snapshot can be used to restart a tenant database at the point in time that the fallback snapshot was created. There is only one fallback snapshot per tenant database. For more information about fallback snapshots, see the SAP HANA Tenant Databases Guide.

**key_management_spec_list**

Defines the key management operations that you can set.

```
<key_management_spec_list> ::=   <add_key_management>   | <update_key_management>   | <activate_key_management>   | <drop_key_management>
```

**add_key_management**

Add a key management configuration for a specific tenant database.

```
<add_key_management> ::=  ADD KEY MANAGEMENT CONFIGURATION <config_name> PROPERTIES '<settings>'
```

- **config_name** Speaks a unique name consisting of uppercase letters only, for example, ‘AWS_HSM’. The name ‘DEFAULT’ is forbidden.
- **settings** Creates a JSON document with key-value settings. The list of keys and their supported values depends on the chosen KMS or HSM. The command will fail if the specified settings do not work.

**update_key_management**

Alters a key management configuration for a specific tenant database.

```
<add_key_management> ::=  ALTER KEY MANAGEMENT CONFIGURATION <config_name> PROPERTIES '<updates>'
```
updates

Modifies a JSON document with the new key-value settings. The list of keys and their supported values depends on the chosen HSM or KMS. All settings not described in the update remain unchanged.

activate_key_management Activate a key management configuration for a specific tenant database.

<activate_key_management> ::= ACTIVATE KEY MANAGEMENT CONFIGURATION <config_name>

drop_key_management

Remove a key management configuration for a specific tenant database.

<drop_key_management> ::= DROP KEY MANAGEMENT CONFIGURATION <config_name>

UNSET LICENSE ALL

Removes all installed licenses on tenant databases from the system database.

Description

Lets you make changes to the database. Resetting the system user password is an audited action on both systemdb and on the tenant. Also, the tenant database must be shutdown before resetting the system user password.

Removing a service by using the ALTER DATABASE statement does not cause the remaining services to be re-distributed automatically.

Permissions

The use of this statement requires the DATABASE ADMIN system privilege.

To use the <key_management_spec_list> clause, you must also have the ENCRYPTION ROOT KEY ADMIN system privilege.

Examples

Add the service scriptserver to the tenant database my_database at hostA, port 30303.

ALTER DATABASE my_database ADD 'scriptserver' AT LOCATION 'hostA:30303';

Change the port number for service scriptserver:

ALTER DATABASE my_database ALTER 'scriptserver' AT 'hostA:30303' TO '30340';
Set the OS user for tenant database `my_database`:

```sql
ALTER DATABASE my_database OS USER 'KPC28' OS GROUP 'admin';
```

Finalize the replica for tenant database `my_database` and drop the source database:

```sql
ALTER DATABASE my_database FINALIZE REPLICA DROP SOURCE DATABASE;
```

Cancel the replica for tenant database `my_database`:

```sql
ALTER DATABASE my_database CANCEL REPLICA;
```

Set the system user password on tenant database `my_database` to `fhu37656`:

```sql
ALTER DATABASE my_database SYSTEM USER PASSWORD fhu37656;
```

Set the restart mode for tenant database `my_database` to not restart after a system restart:

```sql
ALTER DATABASE my_database NO RESTART;
```

Set control of encryption configuration to tenant database `my_database`:

```sql
ALTER DATABASE my_database ENCRYPTION CONFIGURATION CONTROLLED BY LOCAL DATABASE;
```

Turn on log file encryption for tenant database `my_database`:

```sql
ALTER DATABASE my_database LOG ENCRYPTION ON;
```

Drop the fall back snapshot for tenant database `my_database`:

```sql
ALTER DATABASE my_database DROP FALLBACK SNAPSHOT;
```

**Related Information**

- Monitoring and Managing Tenant Databases
- Create a Fallback Snapshot
- `ALTER SYSTEM START DATABASE Statement (Tenant Database Management)` [page 583]
- `M_SNAPSHOTS System View` [page 2134]
- `M_DATABASES System View` [page 1846]
4.10.1.6 ALTER FULLTEXT INDEX Statement (Data Definition)

Changes the parameters of a fulltext index or the state of an index processing queue.

Syntax

```
ALTER FULLTEXT INDEX <index_name> <alter_fulltext_index_option>
```

Syntax Elements

index_name

Specifies the name of the fulltext index to be rebuilt with an optional schema name.

<index_name> ::= [ <schema_name>.]<identifier>
<schema_name> ::= <unicode_name>

alter_fulltext_index_option

Specifies whether the parameters of the fulltext index or the state of the fulltext index queue should be changed.

<alter_fulltext_index_option> ::=   <fulltext_parameter_list>   | <queue_command> QUEUE

Specifying <queue_command> QUEUE is only possible for an asynchronous explicit fulltext index.

fulltext_parameter_list

Specifies a list of fulltext index parameters that are to be changed.

<fulltext_parameter_list> ::=   <fulltext_parameter> [, <fulltext_parameter> [,...] ]

fulltext_parameter

 Specifies fulltext index parameters.

<fulltext_parameter> ::=   FUZZY SEARCH INDEX <on_off>
                          PHRASE INDEX RATIO <index_ratio>
                          TEXT MINING <on_off>
                          TEXT MINING CONFIGURATION <string>
                          TEXT MINING CONFIGURATION OVERLAY <string>

FUZZY SEARCH INDEX on_off

 Defines if a fuzzy search index is used.

FUZZY SEARCH INDEX <on_off>
ON switches on the fuzzy search index, and OFF turns the fuzzy search index off.

**PHRASE INDEX RATIO index_ratio**

Specifies the phrase index ratio.

```
PHRASE INDEX RATIO <index_ratio>
```

index_ratio

Specifies the percentage of the phrase index ratio.

```
<index_ratio> ::= <exact_numeric_literal>
```

The value used must be between 0.0 and 1.0

**queue_command**

Specifies an action to be performed on the index queue.

```
<queue_command> ::= FLUSH | SUSPEND | ACTIVATE
```

**FLUSH**

Updates the fulltext index with the documents in the queue that have already been processed.

**SUSPEND**

Suspends the fulltext index processing queue.

**ACTIVATE**

Activates the fulltext index processing queue.

**TEXT MINING on_off**

Enables/disables text mining on the indexed column.

```
TEXT MINING <on_off>
```

Text mining initialization will be called when the fulltext index is created.

**TEXT MINING CONFIGURATION string**

Specifies the text mining configuration name to use.

```
TEXT MINING CONFIGURATION <string>
```

Also invokes text mining initialization.

**TEXT MINING CONFIGURATION OVERLAY string**

Specifies certain text mining parameters to override the text mining configuration.

```
TEXT MINING CONFIGURATION OVERLAY <string>
```

Also invokes text mining initialization.
Description

The queue is a mechanism used to enable a fulltext index to operate in asynchronous manner; that is, inserts do not block until a document is processed.

Examples

Create table T and then add two indexes i1 and i2. Index i1 is a synchronous fulltext index, i2 is an asynchronous fulltext index.

```
CREATE COLUMN TABLE T (c1 NVARCHAR(1000), c2 NVARCHAR(1000));
CREATE FULLTEXT INDEX i1 ON T (c1) SYNC;
CREATE FULLTEXT INDEX i2 ON T (c2) ASYNC;
```

Alter the phrase index to 30 percent and turn on the fuzzy search index for indexes i1 and i2.

```
ALTER FULLTEXT INDEX i1 PHRASE INDEX RATIO 0.3 FUZZY SEARCH INDEX ON;
ALTER FULLTEXT INDEX i2 PHRASE INDEX RATIO 0.3 FUZZY SEARCH INDEX ON;
```

Suspend the queue for fulltext index i2.

```
ALTER FULLTEXT INDEX i2 SUSPEND QUEUE;
```

Update the fulltext index with the documents in the queue that have already been processed.

```
ALTER FULLTEXT INDEX i2 FLUSH QUEUE;
```

Related Information

SAP HANA Text Mining Developer Guide

4.10.1.7 ALTER FUNCTION Statement (Procedural)

Modifies a user-defined function.

Syntax

```
ALTER FUNCTION <function_name>
{ | ( ( <input_parameter_clause> ) )
  [ RETURNS <return_type> ]
  [ LANGUAGE <lang> ]
  [ SQL SECURITY { DEFINER | INVOKER } ]
```
Syntax Elements

function_name

Specifies the identifier of the function to be altered, with optional schema name.

input_parameter_clause

Specifies the input parameters for the function.

Scalar user-defined functions support primitive SQL types, as well as table parameters (actual physical tables), views, and table variables. Table user-defined functions support table types as input.

sql_type [ ARRAY ]

Specifies the SQL type. Use the optional ARRAY keyword with sql_type if the parameter is an array.

Scalar user-defined functions do not support ALPHANUM, VARBINARY, CLOB, NCLOB, or BLOB types.

table_type
Specifies a table type that was previously defined with the CREATE TYPE statement.

\[
<table_type> ::= [ <schema_name>.]<identifier> \\
/schema_name> ::= <unicode_name>
\]

**table_type_definition**

Specifies a table type that is implicitly defined within the signature.

\[
<table_type_definition> ::= TABLE (<column_list_definition>)  \\
/column_list_definition> ::= <column_elem>[, <column_elem> [,..] ]  \\
/column_elem> ::= <column_name> <data_type>  \\
/column_name> ::= <identifier>
\]

**any_table_type**

Defines the table type at DDL time or query compilation time.

\[
(any_table_type> ::= TABLE(…)
\]

The syntax of \(any_table_type\) requires you to explicitly specify TABLE(…) (for example, CREATE FUNCTION anyfunc_in (IN itab TABLE(…), outtab TABLE(…)) AS BEGIN…).

**RETURNS return_type**

Specifies the return type.

\[
<return_type> ::= { <return_parameter_list> | <return_table_type> }
\]

Scalar functions must return scalar values specified in <return_parameter_list>. Table functions must return a table whose type is defined by <return_table_type>.

**return_parameter_list**

Specifies the output parameters. Use the optional ARRAY keyword for <sql_type> if the return parameter is an array.

\[
<return_parameter_list> ::= <return_parameter>[, <return_parameter> [,..] ]  \\
<return_parameter> ::= <parameter_name> <sql_type> [ ARRAY ]
\]

**return_table_type**

Specifies the structure of the returned table data.

\[
(return_table_type> ::= TABLE ( <ret_column_list> )
\]

**ret_column_list**

Specifies the list of columns returned from the function.

\[
(ret_column_list> ::= <ret_column_elem>[, <ret_column_elem> [,..] ]  \\
(ret_column_elem> ::= <column_name> <sql_type>  \\
/column_name> ::= <identifier>
\]

**LANGUAGE lang**
Specifies the programming language used in the function.

```
LANGUAGE <lang>
<lang> ::= SQLSCRIPT
```

You can only use SQLScript functions.

```
SQL SECURITY { DEFINER | INVOKER }
```

Specifies the security mode of the function.

**DEFINER**

Specifies that the function is executed with the privileges of the definer of the function. DEFINER is the default for table user-defined functions.

**INVOKER**

Specifies that the function is executed with the privileges of the invoker of the function. Invoker is the default for scalar user-defined functions.

```
default_schema_name
minute_value
```

Specifies the result cache retention period.

```
<minute_value> ::= <unsigned_integer>
```

Only stale data access that does not exceed the specified RETENTION period is allowed. For outdated data that exceeds the RETENTION period, internally the result cache is refreshed and up-to-date data is returned.

Specifies the schema for unqualified objects in the function body.

```
<default_schema_name> ::= <identifier>
```

If `<default_schema_name>` is not specified, then the `<current_schema>` of the session is used.

**DETERMINISTIC**

DETERMINISTIC applies to scalar-functions. Signifies that the SQLScript function returns the same result any time it is called with the same input parameters.

**WITH ENCRYPTION**

Encrypts the definition of the function. You cannot decrypt the definition later.

**variable_cache_clause**

Modifies variable cache settings.

```
<variable_cache_clause> ::= {  
   <add_variable_cache_option>  
    | <alter_variable_cache_option>
    | <drop_variable_cache_option>
  }
```

**add_variable_cache**

Adds variable cache settings.

```
<add_variable_cache> ::= ADD VARIABLE CACHE ON {  
   <variable_entry_with_mode_list>  
    | <variable_entry_without_mode_list>  
  }
```
alter_variable_cache

Alters variable cache settings.

<alter_variable_cache> ::= ALTER VARIABLE CACHE ON
<variable_entry_with_mode_list> ::= <variable_entry_with_mode> [, <variable_entry_with_mode> [,...] ]
<variable_entry_with_mode> ::= <variable_entry> <enable_mode>
<variable_entry> ::= { <variable_name> | <variable_name_clustered_list> }
<variable_name_clustered_list> ::= ( <variable_name> [, <variable_name> [,...] ] )
<enable_mode> ::= { ENABLE | DISABLE }

drop_variable_cache

Drops one or all variable cache settings.

<drop_variable_cache> ::= DROP VARIABLE CACHE { ON <variable_list> | ALL }
<variable_list> ::= <variable_name> [, <variable_name> [,...] ]

statement_body

Defines the main body of the function. See the CREATE FUNCTION statement for more information on what <statement_body> can contain. CREATE FUNCTION Statement (Procedural) [page 752]

Description

The result cache stores query execution results on the table function to improve query performance when a query is subsequently executed on the same table. With this ALTER FUNCTION command, the result cache can be added to the table function.

Use ALTER FUNCTION <function_name> ENCRYPTION ON to encrypt the contents of a function without changing <statement_body>. You cannot undo this change.

See the SAP HANA SQLScript Reference guide for more information about creating and modifying user-defined functions.

Examples

Create a table, Tab, and table function function_a that selects all records from table Tab.

CREATE ROW TABLE Tab (COL1 INT PRIMARY KEY, COL2 INT);
CREATE FUNCTION function_a RETURNS TABLE (COL1 INT, COL2 INT) AS
BEGIN
  RETURN SELECT * FROM Tab;
Add a static result cache to the function_a function with a retention of 10:

```
ALTER FUNCTION function_a ADD STATIC CACHE RETENTION 10;
```

Change the retention value of the result cache of the function_a function to 20.

```
ALTER FUNCTION function_a ALTER CACHE RETENTION 20;
```

Restrict the cache entry location for function_a to the (fictitious) indexserver lbsrv18:31576.

```
ALTER FUNCTION function_a ALTER CACHE ADD AT LOCATION 'lbsrv18:31576';
```

Remove the location specifications so that there are no restrictions on the cache entry location for function_a.

```
ALTER FUNCTION function_a ALTER CACHE DROP AT LOCATION 'lbsrv18:31576';
```

Drop the result cache of table function function_a.

```
ALTER FUNCTION function_a DROP CACHE;
```

### Related Information

- CREATE FUNCTION Statement (Procedural) [page 752]
- ALTER FUNCTION
- CREATE FUNCTION
- M_SQLSCRIPT_VARIABLE_CACHE System view [page 2142]
- Predicates [page 61]
- CREATE TYPE Statement (Procedural) [page 892]

### 4.10.1.8 ALTER INDEX Statement (Data Definition)

Alters an index.

### Syntax

```
ALTER INDEX <index_name>
   [   REBUILD
       UNIQUE <cs_inverted_type_index>
       <load_unit>  ]
```
### Syntax Elements

**index_name**

Specifies the name of the index to be rebuilt with an optional schema name.

\[
<\text{index}\_\text{name}> ::= [<\text{schema}\_\text{name}>.]<\text{identifier}> \\
<\text{schema}\_\text{name}> ::= <\text{unicode}\_\text{name}>
\]

**REBUILD**

Rebuilds the index.

**UNIQUE cs_inverted_type_index**

Changes the index type of a composite UNIQUE INVERTED index on a column store table.

\[
<\text{cs}\_\text{inverted}\_\text{type}\_\text{index}> ::= \text{INVERTED} \{ \text{HASH} | \text{VALUE} | \text{INDIVIDUAL} \}
\]

**INVERTED HASH**

INVERTED HASH should not be used as a composite type in cases where range queries or similar queries on the composite keys are a significant part of the workload. In these cases, use INVERTED VALUE instead.

**Note**

Non-unique INVERTED HASH indexes are deprecated. Only UNIQUE INVERTED HASH indexes are supported.

**INVERTED VALUE**

INVERTED VALUE is the default index type for column store tables.

**INVERTED INDIVIDUAL**

An INVERTED INDIVIDUAL index is a lightweight index type for column store tables with reduced memory footprint. The name INVERTED INDIVIDUAL reflects that internally only inverted indexes on individual columns are created (that is, no concat attributes are created). This type of index is only available for multi-column unique constraints, which may be defined as secondary unique index or primary key.

**load_unit**

Changes the load unit for a composite inverted (value or hash) index on a column store table.

\[
<\text{load}\_\text{unit}> ::= \{ \text{COLUMN} | \text{PAGE} | \text{DEFAULT} \} \text{LOADABLE}
\]

**COLUMN LOADABLE** The query results are loaded by column. This is the default for column store tables.

**PAGE LOADABLE** The query results are loaded by page.

**DEFAULT LOADABLE** If you specify DEFAULT, then there is no explicit preference for the load unit that is used when the results are loaded from the table. The index inherits the load unit setting from the first level above it (if specified): column, table, database. If there is no load unit setting to inherit, the default behavior is COLUMN LOADABLE.
Column store inverted indexes can be as large as the column itself. The `<load_unit>` setting of an index impacts the memory footprint of the index. For example, a PAGE LOADABLE index consumes much less memory, but may degrade performance slightly. This may be suitable for an index that is not frequently used. When performance is the priority, you may want to set the index to COLUMN LOADABLE, assuming the memory overhead is acceptable.

The load unit for an index is noted in the LOAD_UNIT column of the INDEXES system view.

**Description**

Alters the specification for an existing index.

**Example**

The following example creates table A1 and an INDEX_1 on column b of table A1:

```
CREATE TABLE A1 (a INT, b NVARCHAR(10), c NVARCHAR(20));
CREATE INDEX_1 ON A1(b);
```

The following example rebuilds INDEX_1:

```
ALTER INDEX_1 REBUILD;
```

The following example alters INDEX_1 to use a target index type INVERTED HASH:

```
ALTER INDEX_1 UNIQUE INVERTED HASH;
```

The following example changes the load unit type for the myIDX index to PAGE LOADABLE:

```
ALTER INDEX myIDX PAGE LOADABLE;
```

**Related Information**

CREATE INDEX Statement (Data Definition) [page 762]
4.10.1.9 ALTER JWT PROVIDER Statement (Access Control)

Alters a JWT provider in the SAP HANA database.

Syntax

```
ALTER JWT PROVIDER <jwt_provider_name>    
{ SET { [ <issuer_clause> ]            
         | [ <claims_clause> ]           
         | [ ... ]        }    
         | <case_clause>      
         | UNSET <unset_claims_clause>     
         | <enable_disable_user_creation_clause>   }
```

Syntax Elements

**jwt_provider_name**

Specifies the identifier of a JWT provider to be modified.

```
<jwt_provider_name> ::= <simple_identifier>
```

**issuer_clause**

Alters the issuer set for the provider.

```
<issuer_clause> ::= WITH ISSUER <issuer_claim>  
<issuer_claim> ::= <string_literal>  
```

*<issuer_claim>* must be unique across all JWT providers.

**claims_clause**

```
<claims_clause> ::=      
[ <external_id_clause> ]     
[ <application_user_clause> ]     
[ <compare_claim_clause> ]
```

**external_id_clause**

Alters the claim set in the JWT tokens to use for mapping the HANA user to an external user name. Once set, the *<external_id_clause>* cannot be unset.

```
<external_id_clause> ::= CLAIM <string_literal> AS EXTERNAL IDENTITY
```

**application_user_clause**

The value is used to set the XS_APPLICATIONUSER session variable during login.

```
<application_user_clause> ::= CLAIM <string_literal> AS APPLICATION USER
```
**compare_claim_clause**

A claim can only be used for one compare operation, be it = or HAS MEMBER.

```plaintext
<compare_claim_clause> ::= { <claim_equals_clause>, ..., <claim_has_member_clause> }, ... }
```

**claim_equals_clause**

<claim_equals_clause> expects the JWT claim value to be a string, number, Boolean, or array (with a single string) and the value must exactly match the configured value (for example, 'origin' claim or 'zone_uuid' claim).

```plaintext
<claim_equals_clause> ::= CLAIM <string_literal> = <string_literal>
```

**claim_has_member_clause**

<claim_equals_clause> expects the JWT claim to be an array of strings. If the JWT, which is used for authentication, contains a simple string instead of an array of strings, then the simple string is treated like an array with one entry. Thus, the effect would be like an equals comparison.

```plaintext
<claim_has_member_clause> ::= CLAIM <string_literal> HAS MEMBER <string_literal>
```

**priority_clause**

Alters the priority order set in which the issuer is checked.

```plaintext
<priority_clause> ::= PRIORITY <number>
```

<number> is a value between 1-255.

Providers with the same issuer must have different priorities. During authentication, the provider with the highest priority is checked first. If the claims match with the provided JWT token, that provider is used for the authentication; otherwise, the provider with the next lower priority is tested until a match is found.

**case_clause**

Specifies whether the user mapping is checked case sensitive (default) or insensitive.

```plaintext
<case_clause> ::= CASE { SENSITIVE | INSENSITIVE } IDENTITY 
```

**unset_claims_clause**

Removes the claims set.

```plaintext
<unset_claims_clause> ::= <unset_claim> [ <unset_claim> ] ...
```<unset_claim> ::= CLAIM <string_literal>

**enable_disable_user_creation_clause**

Enables or disables automatic user creation for JWT authentication.

```plaintext
<enable_disable_user_creation_clause> ::= { ENABLE USER CREATION [ USER TYPE { STANDARD | RESTRICTED } ]
   [ USERGROUP <usergroup_name> ] LDAP AUTHORIZATION
   | DISABLE USER CREATION
}
```
When enabling automatic user creation, optionally specify the type of the new users to be created. By default, new users are created as STANDARD users. The user is created in the specified user group. The RESTRICTED user created using this clause is not automatically granted the PUBLIC role.

Description

For the SET clause, at least one of the two optional parameters must be specified.

Permissions

Only database users that have the USER ADMIN system privilege can alter a JWT provider.

To use the <enable_disable_user_creation_clause> also requires the OPERATOR object privilege on the referenced USERGROUP.

Examples

Change the issuer of the y_jwt_provider JWT provider to www/url/my_new_url.

```
ALTER JWT PROVIDER my_jwt_provider SET ISSUER 'www/url/my_new_url';
```

Change the external identity of the y_jwt_provider JWT provider to user2.

```
ALTER JWT PROVIDER my_jwt_provider SET CLAIM 'user2' AS EXTERNAL IDENTITY;
```

Change the y_jwt_provider JWT provider to be case insensitive.

```
ALTER JWT PROVIDER my_jwt_provider CASE INSENSITIVE IDENTITY;
```

Related Information

CREATE JWT PROVIDER Statement (Access Control) [page 767]
DROP JWT PROVIDER Statement (Access Control) [page 959]
JWT_PROVIDERS System View [page 1589]
JWT_USER_MAPPINGS System View [page 1591]
4.10.1.10 ALTER LDAP PROVIDER Statement (Access Control)

Updates an LDAP provider for use with LDAP authorization and authentication.

Syntax

```sql
ALTER LDAP PROVIDER <ldap_provider_name>    
  [ CREDENTIAL TYPE <credential_type_name> USING <credential_of_ldap_account> ]       
  [ USER LOOKUP URL <url_string_literal> ]    
  [ NESTED GROUP LOOKUP URL { <url_string_literal> | NULL } ]    
  [ ATTRIBUTE MEMBER_OF { <member_of_string_literal> | NULL } ]   
  [ ATTRIBUTE DN <dn_string_literal> ]   
  [ ATTRIBUTE MEMBER_OF { <member_of_string_literal> | NULL } ]  
  [ SSL { ON | OFF } ]   
  [ DEFAULT { ON | OFF } ]   
  [ { ENABLE | DISABLE } PROVIDER ]  
  [ { ENABLE USER CREATION FOR LDAP <usertype_option> | DISABLE USER CREATION FOR LDAP } ]
```

Syntax Elements

ldap_provider_name

Specifies the identifier of a valid LDAP provider.

```sql
<ldap_provider_name> ::= <unicode_name>
```

CREDENTIAL TYPE credential_type_name USING credential_of_ldap_account

Specifies the credential type for the LDAP access account. Only one credential type can be configured for an LDAP provider and the only supported type is PASSWORD.

```sql
<credential_type_name> ::= 'PASSWORD'
<credential_of_ldap_account> ::=  
'user=<user_dn_string_literal>;password=<passphrase>'
```

LDAP access account information includes the distinguished name (DN) and password of the user that is set up in LDAP server for use by SAP HANA server. This user must have permissions within the LDAP server to perform searches as specified by the USER LOOKUP URL and NESTED GROUP LOOKUP URL clauses.

PASSWORD specifies an LDAP access account for which you specify the credentials by using the USING clause.

```sql
<credential_of_ldap_account> specifies the credential of the LDAP access account that SAP HANA uses to log in to the LDAP server.
```

```sql
$user_dn_string_literal> specifies the distinguished name (DN) of the LDAP access account that HANA uses to log in to the LDAP server.
```
<passphrase> specifies the password for the access account that SAP HANA uses to log in to the LDAP server.

An example CREDENTIAL clause might look like this: 'PASSWORD' USING 'user=cn=LookupAccount,o=largebank.com;password=secret'.

USER LOOKUP URL url_string_literal

Specifies an LDAP URL that locates a unique user entry in the LDAP server that corresponds to an SAP HANA user. The format of a USER LOOKUP URL is as follows:

```plaintext
```

A search filter <filter> uses the USER_NAME of the SAP HANA user to locate the user entry in the LDAP Server. <filter> must include a condition of the form '<attr>=*' where <attr> is an LDAP attribute whose value is matched against the USER_NAME of the SAP HANA user. SAP HANA replaces the '*' in the search filter with the USER_NAME of the current SAP HANA user.

For example, when the following statement is executed, (&(objectClass=user) (sAMAccountName=*)) is replaced with (&(objectClass=user) (sAMAccountName=<USER_NAME of the current or specified SAP HANA user>):

```plaintext
USER LOOKUP URL 'ldap://myhostname:389/ou=Users,dc=largebank,dc=com??sub?(member:1.2.840.113556.1.4.1941:=*)'
```

<filter> must contain one and only one '*': otherwise, an error is returned.

The <attributes> specification lists LDAP attributes whose values are returned to the SAP HANA server for a given user entry. <attributes> must be left empty for USER LOOKUP URL. SAP HANA constructs the <attributes> internally based on the LDAP attributes specified with ATTRIBUTE clause.

The following is an example of a USER LOOKUP URL:

NESTED GROUP LOOKUP URL url_string_literal

Specifies an LDAP URL for obtaining LDAP group membership information, including nested groups, for a user from the LDAP server. For example: NESTED GROUP LOOKUP URL 'ldap://myhostname:389/ ou=groupsOU,dc=x??sub?(member:1.2.840.113556.1.4.1941:=*)'

The asterisk in <url_string_literal> is replaced by the user DN, which is obtained by using the USER LOOKUP URL.

Depending upon the NESTED GROUP LOOKUP URL, one or more levels of LDAP groups may be returned. It is possible to specify a URL that returns only one level of groups with NESTED GROUP LOOKUP URL. SAP HANA sends this LDAP query to the LDAP server for execution. Before sending the query, SAP HANA replaces '*' in the NESTED GROUP LOOKUP URL with the DN of the user obtained from execution of USER LOOKUP URL.

While ATTRIBUTE MEMBER_OF and NESTED GROUP LOOKUP URL are both optional, if LDAP authorization is used and neither of these are defined, then an error is returned at run-time.

Support for nested LDAP groups is provided only when a single URL can be specified such that it returns a complete list of groups with the user’s membership (including groups with indirect membership). SAP HANA does not recursively fetch nested groups if they were not obtained by executing the search specified by NESTED GROUP LOOKUP URL.

ATTRIBUTE DN dn_string_literal
Specifies the LDAP attribute that provides the DN (distinguished name) of the LDAP User entry. The DN of the user is stored in the SAP HANA catalog.

**ATTRIBUTE MEMBER_OF member_of_string_literal**

Specifies or removes (NULL) the LDAP attribute that provides a list of groups that a user is a member of. If NESTED GROUP LOOKUP URL is specified, then the group information is obtained using NESTED GROUP LOOKUP URL (that is, ATTRIBUTE MEMBER_OF is not used).

While ATTRIBUTE MEMBER_OF and NESTED GROUP LOOKUP URL are both optional, if LDAP authorization is used and neither of these are defined, then an error is returned at run-time.

**SSL {ON | OFF}**

Specifies whether SSL/TLS secures connections to the LDAP server, both for LDAP access account authentication and LDAP user and group searches. The default is OFF.

⚠️ **Caution**

If using the LDAP server for user authorization and authentication, you must secure communication between SAP HANA and the LDAP server using the TLS/SSL protocol to protect the transmission of sensitive information such as user names, passwords, and group membership, which are otherwise sent in the clear between SAP HANA and the LDAP server.

When using SSL protocol, the trust store used to authenticate communication must be a certificate collection in the SAP HANA database with the purpose LDAP. The certificate of the Certificate Authority (CA) that signed the certificate used by the LDAP server must be available in this certificate collection. For more information, see the section on certificate management in the SAP HANA Security Guide.

When set to ON, the SSL/TLS protocol is used and the URL begins with "ldap://".

Connections to LDAP Server can also be secured by using Secure LDAP protocol. In this case, URL begins with "ldaps://". SSL should be set to OFF when using Secure LDAP protocol.

**DEFAULT {ON | OFF}**

Designates the LDAP provider to use for LDAP authorization and authentication. The default is OFF.

If you create multiple named LDAP providers, then you can designate one as the default. Designating an LDAP provider as DEFAULT removes the default designation of the previous default LDAP provider, if any.

**{ENABLE | DISABLE } PROVIDER**

Enables or disables the use of an LDAP provider. The default is DISABLE PROVIDER.

If a default LDAP provider is disabled, then users cannot log in by using LDAP authorization. VALIDATE LDAP PROVIDER can still be used to verify the configuration of a disabled LDAP provider.

**{ ENABLE | DISABLE } USER CREATION FOR ...**

Enables or disables automatic user creation for LDAP authentication.

```
[ { ENABLE USER CREATION FOR LDAP [ <usertype_option> ] | DISABLE USER CREATION FOR LDAP } ]
<usertype_option> ::= USER TYPE { STANDARD | RESTRICTED }
```

When enabling automatic user creation, optionally specify the type of the new users to be created. By default, new users are created as STANDARD users.
Description

Use the ALTER LDAP PROVIDER statement to update the configuration of an LDAP provider:

If the LDAP provider is set to DEFAULT OFF and there is no other default provider that is enabled, or if the LDAP provider is set to DISABLE PROVIDER and there is no other default provider that is enabled, then:

- Any user connections created before the expiration of the value for the ldap_authorization_role_reuse_duration database property succeed.
- Any user connections created after the expiration of the value for the ldap_authorization_role_reuse_duration database property fail.
- Any first time user connections will fail.

When a default and enabled LDAP provider is altered such that it is either no longer a default LDAP provider or it is no longer enabled, the following occurs with respect to authentication and authorization:

- In case of authentication, the connection fails
- In case of authorization, the user connections may succeed. This is because within the user’s role reuse duration, SAP HANA does not contact the LDAP server to validate the user’s role with LDAP server.

Permissions

Only users with the LDAP ADMIN privilege can alter LDAP providers.

Examples

Example 1: Alter the USER LOOKUP URL for an LDAP provider called my_ldap_provider:

```
ALTER LDAP PROVIDER my_ldap_provider
  USER LOOKUP URL 'ldap://myhostname:389/ou=Users,dc=largebank,dc=com??sub? (&(objectClass=user)(sAMAccountName='*))';
```

Example 2: Make the LDAP provider my_ldap_provider the default and enable it:

```
ALTER LDAP PROVIDER my_ldap_provider DEFAULT ON ENABLE PROVIDER;
```

Example 3: Change the NESTED GROUP LOOKUP URL for the LDAP provider my_ldap_provider:

```
ALTER LDAP PROVIDER my_ldap_provider
  NESTED GROUP LOOKUP URL 'ldap://myhostname:389/ou=groupsOU,dc=x?sub?(member:1.2.840.113556.1.4.1941:=-*)';
```

Example 4: Remove the NESTED GROUP LOOKUP URL setting for the LDAP provider my_ldap_provider, and add the ATTRIBUTE MEMBER_OF setting instead:

```
ALTER LDAP PROVIDER my_ldap_provider
  NESTED GROUP LOOKUP URL NULL
  ATTRIBUTE MEMBER_OF 'memberOf';
```
Example 5: Enable automatic user creation for LDAP authentication:

```
ALTER LDAP PROVIDER my_ldap_provider ENABLE USER CREATION FOR LDAP;
```

Example 6: Change the user type for the new users to be auto-created with LDAP authentication to RESTRICTED:

```
ALTER LDAP PROVIDER my_ldap_provider
   ENABLE USER CREATION FOR LDAP USER TYPE RESTRICTED;
```

Example 7: Disable user creation for LDAP authentication:

```
ALTER LDAP PROVIDER my_ldap_provider DISABLE USER CREATION FOR LDAP;
```

Related Information

Certificate Management in SAP HANA

CREATE LDAP PROVIDER Statement (Access Control) [page 772]
DROP LDAP PROVIDER Statement (Access Control) [page 962]
VALIDATE LDAP PROVIDER Statement (Access Control) [page 1195]
LDAP_PROVIDERS_URLS System View [page 1599]
LDAP_PROVIDERS System View [page 1597]
LDAP_USERS System View [page 1599]

4.10.11 ALTER LIBRARY Statement (SQLScript)

Alters a SQLScript user-defined library.

Syntax

```
ALTER LIBRARY [ <schema_name>.]<library_name>
[ LANGUAGE SQLSCRIPT ]
AS BEGIN
   [ <sqlscript_user_defined_library_spec> ]
END;
```

Syntax Elements

`library_name`

Specifies the name of the SQLScript user-defined library.
sqlscript_user_defined_library_spec

Specifies the body of the new SQLScript user-defined library. See the SAP HANA SQLScript Reference for the syntax allowed in a SQLScript language library.

Description

Use this statement to alter a SQLScript user-defined library in an SAP HANA database.

Examples

The following example creates a table, creates a SQLScript user-defined library that operates on the table, and then alters the body of the library, replacing it to be BEGIN END:

```
CREATE TABLE data_table(col1 INT);
CREATE LIBRARY mylib AS BEGIN
   PUBLIC VARIABLE maxval CONSTANT INT = 100;
   PUBLIC PROCEDURE get_data(IN size INT, OUT result table(col1 INT)) AS
      BEGIN
         RESULT = SELECT TOP :size col1 FROM data_table;
      END;
END;
ALTER LIBRARY mylib AS BEGIN END;
```

Related Information

User-Defined Libraries
SAP HANA SQLScript Reference
CREATE LIBRARY Statement (SQLScript) [page 770]
DROP LIBRARY Statement (SQLScript) [page 961]
LIBRARIES System View [page 1600]
LIBRARY_MEMBERS System View [page 1601]

4.10.1.12 ALTER PROCEDURE Statement (Procedural)

Alters a procedure or manually triggers a recompilation of a procedure by generating an updated execution plan.
Syntax

ALTER PROCEDURE <proc_name>
{ [ { <input_output_parameter_clause> } ]
[ LANGUAGE <lang> ]
[ SQL SECURITY { DEFINER | INVOKER } ]
[ DEFAULT SCHEMA <default_schema_name> ]
[ READS SQL DATA ]
[ <variable_cache_clause> ]
[ DETERMINISTIC ]
[ WITH ENCRYPTION ]
[ AUTOCOMMIT DDL { ON | OFF } ]
AS
BEGIN  [ SEQUENTIAL EXECUTION ]
<statement_body>
END }  | ALTER PROCEDURE <proc_name> RECOMPILE
| ALTER PROCEDURE <proc_name> ENCRYPTION ON

Syntax Elements

proc_name
The procedure name, with optional schema name.

<proc_name> ::= [ <schema_name>.]<identifier>
<schema_name> ::= <unicode_name>

input_output_parameter_clause
Specifies the input and output parameters for the procedure.

<input_output_parameter_clause> ::= <parameter> [,...]
<parameter> ::= [ IN | OUT | INOUT ] <param_name> <data_or_table_type>
<data_or_table_type> ::=<sql_type> [ DEFAULT <constant> ] [ ARRAY ]
| <table_type>
| <table_type_definition>
| <any_table_type>

The default <parameter> is IN. Input and output parameters must be explicitly typed; un-typed tables are not supported.

The input and output parameters of a procedure can have any of the primitive SQL types or a table type. INOUT parameters can only be of scalar type.

sql_type [ DEFAULT constant ] [ ARRAY ]
Specifies the data type of the variable.

<sql_type> ::= DATE
| TIME
| SECONDDATE
| TIMESTAMP
| TINYINT
| SMALLINT
| INTEGER
Only for use with IN parameters; use the optional DEFAULT option when you want to specify a default and a constant is correct for the data type. For example, you cannot specify a constant as a default for an ARRAY.

Use the optional ARRAY option when the parameter is an array.

**table_type**

Specifies a table type that was previously defined with the CREATE TYPE statement.

```plaintext
<table_type> ::= [ <schema_name>.]<identifier>  
<schema_name> ::= <unicode_name>
```

**table_type_definition**

Specifies a table type that is implicitly defined within the signature.

```plaintext
<table_type_definition> ::= TABLE (<column_list_definition>)  
<column_list_definition> ::= <column_elem>[,...]  
<column_elem> ::= <column_name> <data_type>  
<column_name> ::= <identifier>
```

**any_table_type**

Defines the table type at DDL time or query compilation time.

```plaintext
<any_table_type> ::= TABLE(...)  
```

The syntax of `<any_table_type>` requires you to explicitly specify `TABLE(...)` (for example, `CREATE PROCEDURE anyproc_in (IN itab TABLE(...), outtab TABLE(...) ) AS BEGIN...`).

**LANGUAGE**

Specifies the programming language that is used in the procedure.

```plaintext
<language> ::= { SQLSCRIPT | R | GRAPH }
```

The default is SQLSCRIPT. Define the language in all procedure definitions.

Specify GRAPH to use graph script, a domain-specific programming language for custom graph algorithms and analytics.

**SQL SECURITY { DEFINER | INVOKER }**

Specifies the security mode for the procedure. The default is DEFINER.

**DEFINER**

Specifies that the procedure is executed with the privileges of the user who defined the procedure.

**INVOKER**

Specifies that the procedure is executed with the privileges of the user who invoked the procedure.
DEFAULT SCHEMA
Specifies the schema for unqualified objects in the procedure body.

DEFAULT_SCHEMA <default_schema_name>
<default_schema_name> ::= <identifier>

If nothing is specified, then the current_schema of the session is used.

DETERMINISTIC
DETERMINISTIC is for use with SQLScript procedures that produce scalar results. It specifies that the procedure returns the same result any time that it is called with the same input parameters.

READS SQL DATA
Specifies that the procedure is read only and side-effect free. That is, the procedure does not make modifications to the database data or its structure, even if the procedure body contains dynamic SQL calls.

The advantage of using this parameter is that certain optimizations are available for read-only procedures.

WITH ENCRYPTION
Encrypts the procedure definition. See the SAP HANA SQLScript Guide for more information on this clause and the impact of encrypting a procedure.

variable_cache_clause
Modifies variable cache settings.

<variable_cache_clause> ::= {<add_variable_cache_option> | <alter_variable_cache_option> | <drop_variable_cache_option>}

add_variable_cache
Adds variable cache settings.

<add_variable_cache> ::= ADD VARIABLE CACHE ON {<variable_entry_with_mode_list> | <variable_entry_without_mode_list>}<variable_entry_with_mode_list> ::= <variable_entry_with_mode> [, <variable_entry_without_mode_list>]<variable_entry_with_mode> ::= <variable_entry enable_mode><variable_entry> ::= {<variable_name> | <variable_name_clustered_list>}<variable_name_clustered_list> ::= ( <variable_name> [, <variable_name>][,...] )<enable_mode> ::= { ENABLE | DISABLE }

alter_variable_cache
Alters variable cache settings.

<alter_variable_cache> ::= ALTER VARIABLE CACHE ON <variable_entry_with_mode_list><variable_entry_with_mode_list> ::= <variable_entry_with_mode> [, <variable_entry_with_mode>][,...] <variable_entry_with_mode> ::= <variable_entry> <enable_mode><variable_entry> ::= {<variable_name> | <variable_name_clustered_list>}<variable_name_clustered_list> ::= ( <variable_name> [, <variable_name>][,...] )<enable_mode> ::= { ENABLE | DISABLE }
drop_variable_cache

Drops one or all variable cache settings.

```
<drop_variable_cache> ::= DROP VARIABLE CACHE { ON <variable_list> | ALL }<variable_list> ::= <variable_name> [, <variable_name> [, ...]]
```

statement_body

Defines the main body of the procedure. See the CREATE PROCEDURE statement for more information on what <statement_body> can contain. CREATE PROCEDURE Statement (Procedural) [page 776]

AUTOCOMMIT DDL { ON | OFF }

Specifies whether to automatically commit DDL statements in the procedure body. Specify ON when automatic commit of DDL statements is required; for example, during administrative operations such as importing data.

This clause is only supported for SQLScript procedures that are not read-only. The default value is OFF.

SEQUENTIAL EXECUTION

Forces sequential execution of the procedure logic. No parallelism takes place.

Description

For the full description of the clauses for the ALTER PROCEDURE statement with respect to SQLScript procedures, including examples, refer to the CREATE PROCEDURE and ALTER PROCEDURE statements in the SAP HANA SQLScript Guide.

Example

Trigger the recompilation of the `my_proc` procedure.

```
ALTER PROCEDURE my_proc RECOMPILE;
```

Example - Using variable cache

Alters an SQLScript procedure to cache variable X:

```
CREATE TABLE test_cache (a INT);
INSERT INTO test_cache VALUES(5);
CREATE OR REPLACE PROCEDURE test_result_cache LANGUAGE SQLSCRIPT AS BEGIN
  x = SELECT * FROM test_cache;
  SELECT * FROM :x;
END;
ALTER PROCEDURE test_result_cache ADD VARIABLE CACHE ON X ENABLE;
```
4.10.1.13 ALTER PSE Statement (System Management)

Modifies a PSE store.

Syntax

```
ALTER PSE <pse_name> { <certificate_clause>
   | <set_own_certificate_clause>
   | <unset_own_certificate_clause>
   | <add_purpose_object_clause>
   | <drop_purpose_object_clause>
   | <public_key_clause> }
```

Syntax Elements

**pse_name**

Specifies the name of the PSE store to be altered.

```
<pse_name> ::= <identifier>
```

**certificate_clause**

Add or drop the certificate(s) to the PSE store.

```
<certificate_clause> ::= { ADD | DROP } CERTIFICATE (     { <certificate_name> [ , <certificate_name> ]...     | <certificate_id> [ , <certificate_id> ] }...)  <certificate_id> ::= <unsigned_integer>  <certificate_name> ::= <identifier>
```

A PSE that is assigned a purpose can only be changed by a user who owns that specific PSE, or a user with the ALTER privilege on this PSE who also has the privilege to assign this PSE this purpose.

**set_own_certificate_clause**
Sets an own certificate and private key.

```xml
<set_own_certificate_clause> ::= SET OWN CERTIFICATE <certificate_content>
<certificate_content> ::= <string_literal>
```

The string `<certificate_content>` may contain one or several certificates in the PEM format and the private key of the own certificate. One of the certificates is the own certificate and the other certificate(s) are intermediate chain certificates completing the trust chain from the own certificate to the root certificate which the peer trusts. The own certificate and the order of the certificates forming a chain can be identified by the issuer or subject relationship. In case the PSE Store already has an own certificate assigned, it is replaced.

**unset_own_certificate_clause** Removes an own certificate and private key.

```xml
<unset_own_certificate_clause> ::= UNSET OWN CERTIFICATE
```

**add_purpose_object_clause**

Specifies the providers, hosts, or remote sources to add to an existing PSE that already has at least one purpose object assigned.

```xml
<add_purpose_object_clause> ::= ADD { PROVIDER <provider_list> | HOST <host_list> | REMOTE SOURCE <remote_source_list> }
<provider_list> ::= <provider>, <provider> [ ,,... ]
<provider> ::= <simple_identifier>
<host_list> ::= <host>, <host> [ ,,... ]
<remote_source_list> ::= <remote_source>, <remote_source> [ ,,... ]
<remote_source> ::= <simple_identifier>
```

**PROVIDER** can only be specified for a PSE with purpose SAML, JWT, or X509. **HOST** can only be specified for a PSE with purpose SSL. **REMOTE SOURCE** can only be specified for a PSE with purpose REMOTE SOURCE.

For JWT, SAML, and X509, the provider must exist before it can be added. See **CREATE JWT PROVIDER Statement**, **CREATE SAML PROVIDER Statement**, and **CREATE X509 PROVIDER Statement**. For **REMOTE SOURCE**, the remote source must exist before it can be added. See **CREATE REMOTE SOURCE Statement**.

Duplicates defined in `<provider_list>`, `<remote_source>`, or `<host_list>` are ignored.

Host names are case insensitive and should exclude the port number (for example, example.sap.corp). The host name format is validated but not its existence in the topology. If duplicate host names are present, the first match from the SNI context list is used. The same host name can be assigned to multiple PSEs.

Multiple providers, remote sources, or hosts can be assigned to a single purpose, but a provider, remote source, or host can only be assigned to one PSE.

When adding a provider, remote source, or host to a PSE, if the purpose provider, remote source, or host already exists for another PSE, the provider, remote source, or host is reassigned to the new PSE. If this results in no remaining purpose providers, remote source, or hosts for the original PSE, the PSE purpose is removed from the original PSE.

Use the **SET PSE statement with the <purpose_object_list> clause**, not the **ALTER PSE statement with the <add_purpose_object_clause> clause** to add or reassign a purpose provider, remote source, or host to an existing PSE that does not have at least one provider, remote source, or host assigned.

**drop_purpose_object_clause**
Specifies the providers, remote source, or hosts to drop from an existing PSE.

```sql
<drop_purpose_object_clause> ::= DROP { PROVIDER <provider_list> | HOST <host_list> | REMOTE SOURCE <remote_source_list> }
<provider_list> ::= <provider> [ , <provider> [ , ... ] ]
<provider> ::= <simple_identifier>
<host_list> ::= <host> [ , <host> [ , ... ] ]
<remote_source_list> ::= <remote_source> [ , <remote_source> [ , ... ] ]
<remote_source> ::= <simple_identifier>
```

PROVIDER can only be specified for a PSE with purpose SAML or JWT. HOST can only be specified for a PSE with purpose SSL. REMOTE SOURCE can only be specified for a PSE with purpose REMOTE SOURCE.

To drop the last purpose provider, remote source, or host from a PSE, use the SET PSE statement without the `<purpose_object_list>` clause.

```sql
public_key_clause
```

Add or drop a public key to or from a PSE store.

```sql
<public_key_clause> ::=  
  { ADD | DROP } PUBLIC KEY ( <public_key_name> [ , <public_key_name> ]...)
<public_key_name> ::= <unsigned_integer>
```

Description

Modifies a PSE store.

Permissions

To use the purpose object clauses on an SSL purpose, you must have the SSL ADMIN system privilege. To use the purpose object clauses on a JWT or SAML purpose, you must have the USER ADMIN system privilege. To use the certificate clauses you must have the ALTER privilege on the PSE object if you are not the owner of the PSE store. To add or drop a REMOTE SOURCE purpose you must have the system privilege CREATE SOURCE.

Examples

The following example shows how to alter an existing PSE store to set a custom certificate along with a certificate trust chain (certificate content shortened):

```sql
ALTER PSE example_pse SET OWN CERTIFICATE
   '-----BEGIN RSA PRIVATE KEY-----
   MIIFDjBAgkqhkiG9w0BAQUAAwIBAgIQIgk/9q9+8g99w0BBQwDgOIS2qgprFqPxECAggA
   .... MANY LINES LIKE THAT ....
   H0ga/iLVwFq3bJ4y9w0BBQwDgOIS2qgprFqPxECAggA
   -----END RSA PRIVATE KEY-----
   -----BEGIN CERTIFICATE-----
   MIIDTXCAKwgAwIBAgIJA2JbClHIAzAIemaGCSqGSG1ob3DQEBQUAMEUxCzAFAgE=
   .... MANY LINES LIKE THAT ....
   B7xxt8BVc69rLe4H15yGqnx77CLSj3mCx2IUXVqRs5mlSbg094NaxsaUYcm0A6Jq==
```
The following example adds the provider provider1 to the mypse1 PSE.

```sql
ALTER PSE pse1 ADD PROVIDER provider1;
```

The following example adds the host host1.sap.corp to the mypse2 PSE.

```sql
ALTER PSE mypse2 ADD HOST 'host1.sap.corp';
```

The following example removes the provider provider1 from the mypse1 PSE.

```sql
ALTER PSE pse1 DROP PROVIDER provider1;
```

The following example removes the host host1.sap.corp from the mypse2 PSE.

```sql
ALTER PSE mypse2 DROP HOST 'host1.sap.corp';
```

The following example adds another remote source to a PSE that has the purpose REMOTE SOURCE.

```sql
ALTER PSE HDB_PSE ADD REMOTE SOURCE HDB_ADM;
```

The following example adds the certificate saml_cert to the PSE store saml_pse.

```sql
ALTER PSE saml_pse ADD CERTIFICATE saml_cert;
```

Related Information

- CREATE JWT PROVIDER Statement (Access Control) [page 767]
- CREATE SAML PROVIDER Statement (Access Control) [page 803]
- CREATE X509 PROVIDER (Access Control) [page 937]
- SAP HANA User Management
- SET PSE Statement (System Management) [page 1167]
- PSE_PURPOSE_OBJECTS System View [page 1621]
- CREATE REMOTE SOURCE Statement (Access Control) [page 799]
- UNSET PSE Statement (System Management) [page 1181]

4.10.1.14 ALTER REMOTE SOURCE Statement (Access Control)

Modifies the configuration of an external data source that is connected to an SAP HANA database.
Syntax

```
ALTER REMOTE SOURCE <remote_source_name> [ <adapter_clause> ] [ <credential_clause> ]
ALTER REMOTE SOURCE <remote_source_name> { <refresh_clause> | <drop_clause> }
ALTER REMOTE SOURCE <remote_source_name> [ <properties_clause> ]
```

Syntax Elements

**remote_source_name**

Specifies the identifier of the remote source.

```
<remote_source_name> ::= <identifier>
```

**adapter_clause**

Specifies the adapter configuration.

```
<adapter_clause> ::= [ ADAPTER <adapter_name> [ CONFIGURATION FILE <configuration_file> ]
CONFIGURATION <connection_info_string> ]
```

**adapter_name**

Specifies the adapter to be used to access the remote data. Query the ADAPTERS system view for the list of available adapter names.

```
<adapter_name> ::= <identifier>
```

**configuration_file**

Specifies the configuration file for the specified adapter.

**connection_info_string**

Specifies the connection parameters for a given adapter.

For an on-premise remote source:

```
<connection_info_string> ::= 
   { 'DSN=<DSN_entry_name>' |
<host_name>:50111 |
   | Driver=<ODBC_library>; ServerNode=<machine_name>:3<instance#>15 } |
   | <failover_server_name>:3<failover_instance_number>15 |
   | ;sessionVariable:<session_variable_name>=? |
   | ;linkeddatabase_mode=optimized } ' |
```

For an SAP HANA service instance remote source:

```
<connection_info_string> ::= 
   { 'DSN=<DSN_entry_name>' |
<host_name>:50111 |
   | Driver=<ODBC_library>; ServerNode=<tenant_endpoint> } |
   | <failover_tenant_endpoint> ]
```
The failover, session variable, and linked database parameters are only supported for SAP HANA remote sources.

**credential_clause**

Specifies the credential configuration.

```
<credential_clause> ::= WITH CREDENTIAL TYPE <credential_type> | DROP CREDENTIAL TYPE [ 'KERBEROS' | 'PASSWORD' ]
```

**credential_type**

Specifies a credential type and the credential.

```
<credential_type> ::= TYPE [ 'KERBEROS' | 'PASSWORD' [ USING <credentials> ] ]
```

**drop_clause**

Delete all linked objects associated with a linked database using the remote source.

```
<drop_clause> ::= DROP LINKED OBJECTS [ CASCADE | RESTRICT ]
```

The CASCADE option drops all the linked tables and dependent objects associated with linked tables. The RESTRICT option drops linked tables are dropped only if there are no dependencies on any of the linked tables. If this option is used and there are dependent objects on a linked table, then an error is raised, and all linked tables are retained. If no drop option is specified, then all internally generated linked tables with no dependencies are dropped. Linked tables with dependencies are retained.

**refresh_option**

Refreshes metadata for a single linked table or all linked objects using the remote source.

```
<refresh_option> ::= REFRESH LINKED { OBJECTS | TABLE <table_name> }<table_name> ::= <database_name>.<schema_name>.<identifier>
```

*<database_name>* is the name of the remote source. *<identifier>* is the name of the table on the remote source.

**properties_clause**

Changes the value of one or more properties of a single remote source.

```
<properties_clause> ::= { SET PROPERTY <prop_definition_list> | UNSET PROPERTY [ ALL | <prop_list> ] }<prop_definition_list> ::= <prop_definition> [, <prop_definition> [ ,... ] ]<prop_definition> ::= <prop_name> = <prop_value> [ <prop_name> = <prop_value> [ ,... ] ]<prop_list> ::= <prop_name> [ , <prop_name> [ ,... ] ]
```

SET replaces the default property value with a new value. UNSET returns the property value to its default value. For a list of properties that can be set for a remote source, see Listing Remote Source Properties in the SAP HANA Administration Guide.
Description

For more information on creating remote sources, see the SAP HANA Administration Guide.

Permissions

This statement requires the ALTER object privilege on the remote source.

Example

This example alters the remote source MY_REMOTE1 to use the HANA adapter, DSN entry MYHANA1, and set a new technical user credential for the source.

```
ALTER REMOTE SOURCE MY_REMOTE1 ADAPTER HANAODBC CONFIGURATION 'DSN=MYHANA1'
    WITH CREDENTIAL TYPE 'PASSWORD' USING 'user="user1";password="Password1"';
```

This example drops the technical user credential for the MY_REMOTE1 remote source.

```
ALTER REMOTE SOURCE MY_REMOTE1 DROP CREDENTIAL TYPE 'PASSWORD';
```

This example clears all internally generated objects associated with remote source MY_REMOTE1 and drops any dependent objects that reference the linked object.

```
ALTER REMOTE SOURCE MY_REMOTE1 DROP LINKED OBJECTS CASCADE;
```

This example refreshes the metadata of all linked objects associated with remote source MY_REMOTE1.

```
ALTER REMOTE SOURCE MY_REMOTE1 REFRESH LINKED OBJECTS;
```

This example refreshes the metadata for remote table MYSHEMA.MYTABLE on remote source MY_REMOTE1.

```
ALTER REMOTE SOURCE MY_REMOTE1 REFRESH LINKED TABLE MY_REMOTE1.MYSHEMA.MYTABLE;
```

This example enables failover, sets the session variable APPLICATIONUSER, and enables optimized mode for linked database on the SAP HANA remote source MY_REMOTE1.

```
ALTER REMOTE SOURCE MY_REMOTE1 ADAPTER HANAODBC
    CONFIGURATION
    'Driver=libodbcHDB.so;ServerNode=my_machine1:30115,failover_machine1:30215
    sessionVariable:APPLICATIONUSER=?;linkeddatabase_mode=optimized']';
```

This example sets two properties on the remote source RS1

```
ALTER REMOTE SOURCE RS1 SET PROPERTY 'CAP_LIMIT' = 'true',
    'PROP_USE_UNIX_MANAGER' = 'true';
```

This example unsets all properties on remote source RS1

```
ALTER REMOTE SOURCE RS1 UNSET PROPERTY ALL;
```
Related Information

Creating Remote Sources
Configure Remote Source Capabilities

4.10.1.15 ALTER ROLE Statement (Access Control)

Adds and drops the mapping of LDAP groups for a role.

Syntax

```
ALTER ROLE <role_name>   
  ADD LDAP GROUP <ldap_group_list>   
  | DROP LDAP GROUP <ldap_group_list>
```

Syntax Elements

- **role_name**
  Specifies the name of the role to be altered with optional schema name.

  ```
  <role_name> ::= [<schema_name>]<identifier>  
  <schema_name> ::= <unicode_name>
  ```

- **ldap_group_list**
  Specifies the Distinguished Name (DN) of one or more LDAP groups.

  ```
  <ldap_group_list> ::= <ldap_group_name> [,...]   
  <ldap_group_name> ::= <string_literal>
  ```

Description

A local role mapped to an LDAP group can also be granted to another role or a user.

For a list of system roles provided in the SAP HANA database, see the GRANT statement. To see all roles, including custom roles that have been created using the CREATE ROLE statement, query the ROLES system view. To view which roles and privileges have actually been granted, query the GRANTED_ROLES and GRANTED_PRIVILEGES system views, respectively.
Permissions

Only database users with the ROLE ADMIN system privilege can alter roles.

Examples

Alter a role associated with an LDAP group:

```sql
ALTER ROLE Securities_DBA
ADD LDAP GROUP 'cn=Securities_DBA,OU=Application,OU=Groups,ou=DatabaseAdmins,cn=Users,o=verylargebank.com';
```

Related Information

- CREATE ROLE Statement (Access Control) [page 802]
- DROP ROLE Statement (Access Control) [page 967]
- ROLES System View [page 1646]
- GRANTED_ROLES System View [page 1581]
- GRANTED_PRIVILEGES System View [page 1580]

4.10.1.16 ALTER SAML PROVIDER Statement (Access Control)

Changes the properties of the specified SAML provider.

Syntax

```sql
ALTER SAML PROVIDER <saml_provider_name> 
{  <subject_issuer_clause>
    | <entityid_clause>
    | <case_clause>
    | <disable_user_creation_clause>
  }
```

Syntax Elements

- `saml_provider_name`
Specifies the SAML provider to alter.

```sql
<saml_provider_name> ::= <simple_identifier>
```

**subjectIssuerClause**

Sets the SAML identity provider information.

```sql
<subjectIssuerClause> ::= SET SUBJECT <subject_distinguished_name> ISSUER <issuer_distinguished_name>
```

**subject_distinguished_name**

Specifies the subject name provided in the certificate of the SAML identity provider.

```sql
<subject_distinguished_name> ::= <string_literal>
```

**issuer_distinguished_name**

Specifies the issuer name provided in the certificate of the SAML identity provider.

```sql
<issuer_distinguished_name> ::= <string_literal>
```

**entityidClause**

Sets or unsets the entity ID of the SAML identity provider.

```sql
<entityidClause> ::= [ UNSET ] ENTITY ID <string_literal>
```

**caseClause**

Specifies whether the user mapping is checked case sensitive (default) or insensitive.

```sql
<caseClause> ::= CASE { SENSITIVE | INSENSITIVE } IDENTITY
```

**enableDisableUserCreationClause**

Enables or disables automatic user creation for SAML authentication.

```sql
<enableDisableUserCreationClause> ::= { ENABLE USER CREATION [ USER TYPE { STANDARD | RESTRICTED } ] [ USERGROUP <usergroup_name> ] [ LDAP AUTHORIZATION ] | DISABLE USER CREATION }
```

When enabling automatic user creation, optionally specify the type of the new users to be created. By default, new users are created as STANDARD users. The user is created in the specified user group. The RESTRICTED user created using this clause is not automatically granted the PUBLIC role.

**Description**

The SAML provider must already exist in the SAP HANA database.
Permissions

Only database users with the USER ADMIN system privilege are allowed to change SAML providers.

To use the `<enable_disable_user_creation_clause>` also requires the OPERATOR object privilege on the referenced USERGROUP.

Example

This example alters the ac_saml_provider.

```
ALTER SAML PROVIDER ac_saml_provider
  SET SUBJECT 'CN = wiki.detroit.BCompany.corp,OU = BCNet,O = BCompany,C = EN'
  ISSUER 'E = John.Do@acompany.com,CN = BCNetCA,OU = BCNet,O = BCompany,C = EN';
```

This example enables user creation and uses the default LOCAL as authorization mode of automatically created users. For more information on authorization modes, see the CREATE USER Statement.

```
ALTER SAML PROVIDER ac_saml_provider ENABLE USER CREATION;
```

This example makes the case identity insensitive.

```
ALTER SAML PROVIDER ac_saml_provider CASE INSENSITIVE IDENTITY;
```

This example removes the entity ID defined for the ac_saml_provider SAML provider.

```
ALTER SAML PROVIDER ac_saml_provider UNSET ENTITY ID;
```

Review the changes made to the ac_saml_provider SAML provider.

```
SELECT * FROM SYS.SAML_PROVIDERS;
```

Related Information

SAML_PROVIDERS System View [page 1648]
CREATE SAML PROVIDER Statement (Access Control) [page 803]
CREATE USER Statement (Access Control) [page 893]
4.10.1.17 ALTER SEQUENCE Statement (Data Definition)

Alters an existing sequence.

Syntax

```sql
ALTER SEQUENCE <sequence_name>    
[ <restart_with >]    
[ <parameter_list> ]    
[ RESET BY <reset_by_subquery> ]
```

Syntax Elements

**sequence_name**

Specifies the name of the sequence to be created, with optional schema name.

```sql
<sequence_name> ::= [ <schema_name>.]<identifier>
```

**restart_with**

Optionally, specifies the starting value of the sequence.

```sql
<restart_with> ::= RESTART WITH <restart_value>                       
<restart_value> ::= <signed_integer>
```

If you do not specify a value for the RESTART WITH clause, then the current value of the sequence is used.

**parameter_list**

Optionally, specifies the sequence parameter list.

```sql
<parameter_list> ::=   <sequence_parameter> [, <sequence_parameter> [...]]
```

**sequence_parameter**

Specifies a set of parameters that can be set.

```sql
<sequence_parameter> ::=    INCREMENT BY <increment_value>   
MAXVALUE <max_value>  
NO MAXVALUE  
MINVALUE <min_value>  
NO MINVALUE  
CYCLE  
NO CYCLE  
CACHE <cache_size>  
NO CACHE
```
INCREMENT BY increment_value

Specifies the amount that the next sequence value is incremented from the last value assigned.

\[
\text{INCREMENT BY } <\text{increment_value}>
\]

<increment_value> must be an integer between -9999999999999999999999999999 to 9999999999999999999999999999 (28 digits) if MINVALUE and/or MAXVALUE is extended. Otherwise, the INCREMENT BY value must be between the default MINVALUE and MAXVALUE range, which is -2^63 to 2^63-1. The default is 1.

Specify a negative value to generate a descending sequence.

An error is returned if the INCREMENT BY value is 0.

MAXVALUE max_value

Specifies the maximum value that can be generated by the sequence.

\[
\text{MAXVALUE } <\text{max_value}>
\]

<max_value> must be a value between -9999999999999999999999999999 and 9999999999999999999999999999 (28 digits). If MAXVALUE is not specified the default MAXVALUE is set as described in the NO MAXVALUE directive.

NO MAXVALUE

When the NO MAXVALUE directive is used, the maximum value for an ascending sequence is 2^63-1 and the maximum value for a descending sequence is -1.

MINVALUE min_value

Specifies the minimum value that a sequence can generate.

\[
\text{MINVALUE } <\text{min_value}>
\]

<min_value> must be a value between -9999999999999999999999999999 and 9999999999999999999999999999 (28 digits). If MINVALUE is not specified the default MINVALUE is set as described in the NO MINVALUE directive.

NO MINVALUE

When the NO MINVALUE directive is used, the minimum value for an ascending sequence is 1 and the minimum value for a descending sequence is -2^63.

CYCLE

When the CYCLE directive is used, the sequence number restarts after it reaches its maximum or minimum value.

NO CYCLE

Specifies the default option. When the NO CYCLE directive is used, the sequence number does not restart after it reaches its maximum or minimum value.

CACHE cache_size

Specifies the cache size for caching sequence numbers in a node.

\[
\text{CACHE } <\text{cache_size}>
\]
<cache_size> ::= <unsigned_integer>

<cache_size> must be an unsigned integer and cannot exceed 2^31-1.

NO CACHE

Specifies to not cache sequence numbers in a node. This is the default behavior.

RESET BY subquery

During the restart of the database, the database automatically executes the <subquery> and the sequence value is restarted with the returned value.

If RESET BY is not specified, then the sequence value is stored persistently in database. During the restart of the database, the next value of the sequence is generated from the saved sequence value.

For more information on subqueries, see the SELECT statement.

Description

See the CREATE SEQUENCE statement for more information about sequence behaviors and how to use sequences.

When altering <cache_size>, the new sequence value will be the former value plus the cache size, resulting in skipped sequence values. The following example demonstrates this behavior:

CREATE SEQUENCE S1 START WITH 10 CACHE 100;
SELECT S1.NEXTVAL FROM DUMMY;  ==> The sequence value returned is 10
ALTER SEQUENCE S1 CACHE 10;
SELECT S1.NEXTVAL FROM DUMMY;  ==> The sequence value returned is 110

Examples

Example 1

Create table A and a sequence seq. Sequence seq, when reset, starts from the value of the select statement shown.

CREATE ROW TABLE A (a INT);
CREATE SEQUENCE seq RESET BY SELECT IFNULL(MAX(a), 0) + 1 FROM A;

Change the starting sequence value of sequence seq to 2.

ALTER SEQUENCE seq RESTART WITH 2;

Change the maximum value of sequence s to 100, and specify that it does not have a minimum value.

ALTER SEQUENCE seq MAXVALUE 100 NO MINVALUE;

Change the incremental value of sequence seq to 3, and specify that the sequence does not restart upon reaching its maximum or minimum value.

ALTER SEQUENCE seq INCREMENT BY 3 NO CYCLE;
Example 2
Create table B, with column a. You create a sequence s1 with a reset-by subquery based on table B.

```sql
CREATE ROW TABLE B (a INT);
CREATE SEQUENCE s1 RESET BY SELECT IFNULL(MAX(a), 0) + 1 FROM B;
```

Change the reset-by subquery of sequence s1 to the maximum value contained in column a of table B.

```sql
ALTER SEQUENCE s1 RESET BY SELECT MAX(a) FROM B;
```

Related Information

- SEQUENCES System View [page 1654]
- M_SEQUENCES System View [page 2105]
- CREATE SEQUENCE Statement (Data Definition) [page 810]
- SELECT Statement (Data Manipulation) [page 1104]

4.10.1.18 ALTER SCHEDULER JOB Statement (Data Definition)

Alters a scheduled job in the current or specified schema.

Syntax

```sql
ALTER SCHEDULER JOB [ <schema_name>.] <job_name>      
[ CRON '<cron>' ] [ <job_recurrence_range> ]     
[ ENABLE | DISABLE ] [ PARAMETERS <parameter_list> ]
```

Syntax Elements

- **job_name** Specifies the name of the scheduled job.
- **cron** Specifies the frequency of the job.

```sql
<cron> ::= <year> <month> <date> <weekday> <hour> <minute> <seconds>
```

- **year** A four-digit number.
- **month** A number from 1 to 12.
date A number from -31 to 31.
weekday A three-character day of the week: mon,tue,wed,thu,fri,sat,sun.
hour A number from 0 to 23 (expressed in 24-hour format).
seconds A number from 0 to 59.

Each cron field also supports wildcard characters as follows.

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Any value.</td>
</tr>
<tr>
<td>*/n</td>
<td>Any n-th value. For example, */1 for the day of the month means run every day of the month, */2 for every second day.</td>
</tr>
<tr>
<td>a:b</td>
<td>Any value between a and b.</td>
</tr>
<tr>
<td>a:b/n</td>
<td>Any n-th value between a and b. For example, 1:10/3 for the day of the month means every 3rd day between 1 and 10 or the 3rd, 6th, and 9th day of the month.</td>
</tr>
<tr>
<td>n.a</td>
<td>(For weekday only) A day of the week where n is a number from -5 to 5 for the n-th occurrence of the day in week a. For example, for the year 2019, 2.3 means Tuesday, January 15th. -3.22 means Friday, May 31st.</td>
</tr>
</tbody>
</table>

job_recurrence_range Specifies the range for the recurrence of the scheduled job. If this value is not specified, then the recurrence of the scheduled job never ends.

\[
<job_recurrence_range> ::= FROM <start_time> \] \[ UNTIL <end_time>
\]

start_time Specifies the earliest time after which the scheduled job can start to run.
end_time Specifies the latest time before which the scheduled job can start to run.
ENABLE | DISABLE Specifies whether the scheduled job will run once the next scheduled time is reached.

parameter_list

Specifies the parameter for the specified procedure, in a comma-separated list of \(<name> = <constant_expression>\).

\[
<parameter_list> ::= <parm_name>=<parm_expression> [, \<parm_name>=<parm_expression> [,...] ] \]

All IN or INOUT parameters without a DEFAULT must be provided.

Permissions

This statement requires that you own the scheduled job or have the ALTER object privilege on the scheduled job.

Example

This example stops scheduling the job schedule1.

ALTER SCHEDULER JOB schedule1 DISABLE;
This example enables the job schedule1, and schedules it to run in any year or month, every other day of the week (*/2) at 12:34:56. It also specifies the value for the DAYS parameter at 7, which is required by the procedure.

```
ALTER SCHEDULER JOB schedule1 CRON "*/2 * * */ * 12 34 56' ENABLE PARAMETERS DAYS=7;
```

**Related Information**

CREATE SCHEDULER JOB Statement (Data Definition) [page 806]
ALTER SCHEDULER JOB Statement (Data Definition) [page 482]

### 4.10.1.19 ALTER STATISTICS Statement (Data Definition)

Alters the properties of a data statistics object.

**Syntax**

```
ALTER STATISTICS { <data_statistics_name> [,]... | ON <data_sources> 
[ [ HAVING ] <match_properties> ] 
[ SET <set_data_statistics_properties> ] 
[ <add_drop_data_statistics_properties> ] 
[ <initial_refresh> ]
```

**Syntax Element**

*data_statistics_name*

Specifies the name of the data statistics object.

```
<data_statistics_name> ::= [<schema_name>.]<identifier>  
/schema_name ::= <identifier>
```

*data_sources*

Specifies the data source(s) of the data statistics objects.

```
<data_sources> ::=  
<table_name> [ ( <column_name>[, <column_name>[,]... ] ) ] [ <match_type> ]
```

For RECORD COUNT data statistics objects, you cannot specify columns as part of `<data_sources>`.

*table_name*
Specifies the table on which the data statistics are defined.

\[
<\text{table\_name}> ::= \begin{cases} [ \begin{cases} \text{<database\_name>} \cdot \text{<schema\_name>} \cdot \text{<identifier>} \end{cases} \end{cases}
\]

For linked database, \(<\text{database\_name}>\) is the name of the remote source. For all other cases, \(<\text{database\_name}>\) is the name of the database where the table is located.

**column\_name**

Specifies the column for which the data statistics are defined.

\[
<\text{column\_name}> ::= \text{<identifier>}
\]

If no \(<\text{column\_name}>\) is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

**match\_type**

Controls which data statistics objects to match to \(<\text{data\_sources}>\).

\[
<\text{match\_type}> ::= \text{EXACT} \mid \text{CASCADE}
\]

If \(<\text{match\_type}>\) is not specified, then any data statistics object(s) that reference all or some of the columns, but no other columns specified in \(<\text{data\_sources}>\) are refreshed.

Specify EXACT to refresh data statistics objects that precisely match \(<\text{data\_sources}>\) (including column order).

Specify CASCADE to refresh data statistics objects that reference at least one column in \(<\text{data\_sources}>\).

Use this table to understand how matching is performed based on \(<\text{match\_type}>\) when \(<\text{data\_sources}>\) is T(A, B, C):

<table>
<thead>
<tr>
<th>Match type</th>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>(not specified)</td>
<td>T(A,C)</td>
<td>T(A,X) · because T(X) is not a column in (&lt;\text{data_sources}&gt;).</td>
</tr>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,A)</td>
<td></td>
</tr>
<tr>
<td>EXACT</td>
<td>T(A,B,C)</td>
<td>T(B,A,C) · because the column order is different than the column order of (&lt;\text{data_sources}&gt;).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T(A) · because it does not contain the exact same columns and column order of (&lt;\text{data_sources}&gt;).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T(X,A,B,C) · because T(X) is not a column in (&lt;\text{data_sources}&gt;).</td>
</tr>
<tr>
<td>CASCADE</td>
<td>T(A,C)</td>
<td>T(X) · because it does not contain any columns that match the columns in (&lt;\text{data_sources}&gt;).</td>
</tr>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,A)</td>
<td></td>
</tr>
</tbody>
</table>
### Match type

<table>
<thead>
<tr>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(A,B,C)</td>
<td></td>
</tr>
<tr>
<td>T(A,X)</td>
<td></td>
</tr>
<tr>
<td>T(B,C)</td>
<td></td>
</tr>
<tr>
<td>T(A)</td>
<td></td>
</tr>
<tr>
<td>T(C,B,A,X)</td>
<td></td>
</tr>
</tbody>
</table>

### match_properties

Specifies properties to use for matching when selecting data statistics.

\[
<\text{match_properties}> ::= <\text{match_property}>[...]
\]

\[
<\text{match_property}> ::=\]

- **TYPE** <data_statistics_type>  
- **REFRESH** TYPE <refresh_type_filter>

If **TYPE** is not specified, then all data statistics objects of any type on the specified data sources are refreshed (ALL). For descriptions of the supported data statistics types see the **CREATE STATISTICS Statement** topic.

### data_statistics_type

Specifies the type of data statistics objects to match when selecting the data statistics.

\[
<\text{data_statistics_type}> ::= \text{TYPE }<\text{type_name}>
\]

\[
<\text{type_name}> ::=\]

- HISTOGRAM
- SIMPLE
- TOPK
- SKETCH
- SAMPLE
- RECORD COUNT
- ALL

### refresh_type_filter

Specifies the refresh strategy to match in the data statistics objects when selecting the data statistics to refresh. ALL is the default.

\[
<\text{refresh_type_filter}> ::= \text{AUTO} \mid \text{MANUAL} \mid \text{ALL}
\]

### set_data_statistics_properties

Specifies the properties of the data statistics objects to modify.

\[
<\text{data_statistics_properties}> ::=<\text{data_statistics_property}>[, <\text{data_statistics_property}>][,...]
\]

\[
<\text{data_statistics_property}> ::=\]

- **REFRESH** TYPE <refresh_type>
- ENABLE <on_off>
- BUCKETS <unsigned_integer>
- QERROR <numeric_literal>
- QTHETA <unsigned_integer>
- \{ MEMORY <memory_bytes> | MEMORY PERCENT <memory_percentage> \}
- ACCURACY <numeric_literal>
- PREFIXBITS <unsigned_integer>
- PERSISTENT <on_off>
REFRESH TYPE `refresh_type`

Specifies the strategy for the data statistics object.

```
<refresh_type> ::= { AUTO | MANUAL | DEFAULT }
```

- `AUTO` specifies that the data statistics object is refreshed automatically when underlying data changes. AUTO is only supported on column store, extended store, and multistore tables.
- `MANUAL` specifies that the database statistics object is not refreshed until a rebuild is explicitly requested by a REFRESH STATISTICS statement.
- `DEFAULT` specifies that the database server decides the best refresh strategy based on the data source. For example, for data statistics objects on column store data sources, the database server applies AUTO for the default.

REFRESH TYPE only affects data statistics objects that are enabled.

**ENABLE `on_off`**

Controls whether the optimizer uses the data statistics object.

```
<on_off> ::= { ON | OFF }
```

- `ENABLE ON` enables the optimizer to see the data statistics object. The data statistics object must be populated with data for the optimizer to use it.
- `ENABLE OFF` disables the use of the data statistics object by the optimizer and prevents the ability to refresh the data statistics object. Data statistics objects that are not enabled can still be dropped. To make a data statistics object with ENABLE OFF accessible to the optimizer, execute an ALTER STATISTICS...ENABLE ON statement.

**BUCKETS `unsigned_integer`**

The BUCKETS property is only for use with TYPE HISTOGRAM or TOPK. For HISTOGRAM, BUCKETS specifies the maximum number of data buckets in the HISTOGRAM. For TOPK, BUCKETS specifies the K value.

The default is automatically determined by the data statistics building algorithm in use.

If a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

For column store, extended store, and multistore tables only, if a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

**QERROR `numeric_literal`**

Specifies the Q error to use for the q-optimal HISTOGRAM. You can specify this parameter when TYPE is HISTOGRAM and CONSTRAINT is QOPTIMAL. The default is automatically determined by the HISTOGRAM algorithm in use.

**QTHETA `unsigned_integer`**

SAP HANA SQL Reference Guide for SAP HANA Platform

SQL Reference

PUBLIC 487
Specifies a lower bound on the frequencies for which a q error constraint is applied for a q-optimal HISTOGRAM. You can specify this parameter when TYPE is HISTOGRAM and CONSTRAINT is QOPTIMAL. The default is automatically determined by the HISTOGRAM algorithm in use.

**MEMORY memory_bytes**

Specifies the maximum amount of memory, in bytes, to use for QOPTIMAL HISTOGRAMS.

\[<\text{memory_bytes}> ::= <\text{unsigned_integer}>\]

The MEMORY parameter limits the memory for QOPTIMAL HISTOGRAMS. MEMORY applies only to the QOPTIMAL HISTOGRAM algorithm. Small values for MEMORY may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

**MEMORY PERCENT memory_percentage**

Specifies the maximum amount of memory to use for the data statistics object, expressed as a percentage of the space used by the data source.

\[<\text{memory_percentage}> ::= <\text{unsigned_integer}>\]

HISTOGRAMS can use a large amount of memory for some data sources. \(<\text{memory_percentage}>\) represents the maximum amount of memory that can be used for the data statistics object. For example, if a data source is a table column that uses 100 MB of memory, and \(<\text{memory_percentage}>\) is 5, then the data statistics object for this column can use, at most, 5 MB for its in-memory representation.

The default is automatically determined by the HISTOGRAM algorithm in use. Small values for MEMORY PERCENT may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

**PERSISTENT on_off**

Specifies whether data statistics object data persists in the storage of the table, and only applies to QOPTIMAL HISTOGRAMS on column store tables. The default is PERSISTENT ON.

\[<\text{on_off}> ::= \{ \text{ON} \mid \text{OFF} \}\]

Other statistics types are always persistent.

**ACCURACY numeric_literal**

Controls the time and space requirements to use for the SKETCH algorithms. This parameter can only be specified when TYPE is SKETCH and must be a number between 0 and 1, with larger values causing decreased time and space requirements but poorer SKETCH resolution. The default is 0.1.

**PREFIX BITS unsigned_integer**

Controls the number of bits the SKETCH algorithms use when constructing the SKETCH statistics. Specify this parameter when TYPE is SKETCH. Its value is an integer between 0 and 63. The default is 8.

**CONSTRAINT constraint_param**

Specifies constraints to use for the specified \(<\text{data_statistics_type}>\). For SKETCH, \(<\text{constraint_param}>\) specifies the algorithm to use to build the SKETCH. The default is LOGLOGCOUNTING; the remaining algorithms are for internal use.

\[<\text{constraint_param}> ::= \]
add_drop_data_statistics_properties

Specifies the properties of the data statistics objects you set using the ADD and DROP keywords.

```sql
<add_drop_data_statistics_properties> ::= <add_drop_property> [ <add_drop_property> […] ]
<add_drop_property> ::= { ADD | DROP } VALID FOR <valid_for_list>
```

This property is supported for column store, extended store, and multistore tables, but not virtual tables or linked database.

**VALID FOR valid_for_list**

Defines how the data statistics object may be used. The VALID FOR clause is only permitted with statistics type SIMPLE.

```sql
<valid_for_list> ::= <usage>[, <usage>[,…]]
<usage> ::= { ESTIMATION | DATA DEPENDENCY }
```

ESTIMATION initializes the data statistics object for use by the optimizer to improve selectivity estimation. SIMPLE data statistics objects are initialized for estimation use by default.

DATA DEPENDENCY applies to partitioned column store and multistore tables only. It initializes the data statistics object to be used by features that require higher (or more) data consistency (including automatically refreshing and rebuilding when needed), such as the dynamic partition pruning feature.

For more information about the dynamic partition pruning feature, including the types of columns that can have data statistics defined for dynamic partition pruning, see the SAP HANA Administration Guide.

**initial_refresh**

Specifies whether to repopulate the data statistics object with data after altering it.

```sql
<initial_refresh> ::= [ NO ] INITIAL REFRESH
```

If the object was built, then disabled, and is now being re-enabled, then initial refresh is not required.

**INITIAL REFRESH**

Alters the definition of the data statistics object and repopulates it with data. The default behavior is INITIAL REFRESH.

**NO INITIAL REFRESH**

Alters the definition of the data statistics object, but does not repopulate it with data.

Use NO INITIAL REFRESH when you want to change the underlying data before refreshing the data statistics object.

You cannot specify NO INITIAL REFRESH if ENABLE OFF is not specified.
Permissions

One of the following is true:
• You own the object you are altering statistics on.
• You have the ALTER privilege on the object you are altering statistics on.
• For linked database, you have the LINKED DATABASE object level privilege on the remote source, regardless of who owns the remote source.

Description

The ALTER STATISTICS statement alters the properties of a data statistics object. A typical change to a data statistics object might be to enable or disable it, or to change settings for properties such as BUCKETS or ENABLE ON/OFF.

ADD, DROP and SET clauses can be specified in any order in the ALTER STATISTICS statement, but not more than once each.

You cannot alter the type for the data statistics object. For example, you cannot change a data statistics object from a HISTOGRAM to a SKETCH, you must create the SKETCH data statistics object separately.

Example

The following example sets the refresh type of a HISTOGRAM on data source T(X) to AUTO and rebuilds the data statistics object:

```
ALTER STATISTICS on MYSYSTEM.T(X) TYPE HISTOGRAM SET REFRESH TYPE AUTO;
```

The following example alters the MYSIMPLESTAT data statistics object to enable it for use with dynamic partition pruning and rebuilds the data statistics object:

```
ALTER STATISTICS MYSIMPLESTA ADD VALID FOR DATA DEPENDENCY;
```

The following example sets the number of buckets to 10 on the remote source REMOTE2 using linked database. The NO INITIAL REFRESH clause means the new setting will not take effect until the next time the object is rebuilt using the REFRESH STATISTICS statement.

```
ALTER STATISTICS on "REMOTE2"."SYSTEM"."A1" TYPE TOPK SET BUCKETS 10 NO INITIAL REFRESH;
```

The following example sets the number of buckets to 150 for the TOPK data statistics object on the virtual table REMOTE2_A1.

```
ALTER STATISTICS on MYSYSTEM.REMOTE2_A1 TYPE TOPK SET BUCKETS 150;
```
Related Information

CREATE STATISTICS Statement (Data Definition) [page 815]
REFRESH STATISTICS Statement (Data Definition) [page 1082]
DROP STATISTICS Statement (Data Definition) [page 974]
M_DATA_STATISTICS System View [page 1851]
M_SYSTEM_DATA_STATISTICS System View [page 2189]
DATA_STATISTICS System View [page 1538]

4.10.1.20  ALTER STRUCTURED PRIVILEGE Statement (Access Control)

Alters a structured (analytic) privilege.

Syntax

```
ALTER STRUCTURED PRIVILEGE <privilege_name> FOR <action>
   ON <object_name_list> <filter_condition>
```

Syntax Elements

```
privilege_name
Specifies the name of the privilege.
<privilege_name> ::= [<schema>.[<identifier>]

action
Specifies the action the privilege allows.
<action> ::= SELECT

object_name_list
Specifies the views to be restricted by the privilege.
<object_name_list> ::= <object_name> [{, <object_name>}...]
<object_name> ::= [<schema>.[<identifier>]

<object_name> specifies a view registered for SQL-based analytic privilege check.

filter_condition
```
Restricts the returned rows of the specified views for users who have been granted the privilege.

<filter_condition> ::= <static_filter_condition> | <dynamic_filter_condition>

<static_filter_condition> ::= WHERE <condition>

<dynamic_filter_condition> ::= CONDITION PROVIDER <procedure_name>

<procedure_name> ::= [<schema>.]<identifier>

<condition> restricts the rows returned and can contain subqueries.
<procedure_name> specifies the procedure providing the dynamic restriction.

Permissions

You must have the STRUCTUREDPRIVILEGE ADMIN privilege to execute this statement.

Description

Alters an analytic privilege. Analytic privileges provide fine-grained control over which data a user can see within a view. Analytic privileges are sometimes referred to as structured privileges.

Related Information

Analytic Privileges
STRUCTURED_PRIVILEGES System View [page 1662]
DROP STRUCTURED PRIVILEGE Statement (Access Control) [page 978]
CREATE STRUCTURED PRIVILEGE Statement (Access Control) [page 822]

4.10.1.21 ALTER SYSTEM ... (All ALTER SYSTEM Statements)
4.10.1.21.1 ALTER SYSTEM {ADD | ALTER | REMOVE} STATEMENT HINT Statement (System Management)

Adds, alters, or removes statement hints from the system to a specified query or statement hash.

**Syntax**

```sql
ALTER SYSTEM
  { ADD STATEMENT HINT ( [ <hint_list> ] ) [ [ NO ] OVERRIDE ] [ COMMENT <comment> ] <statement_hint_target> 
  | ALTER STATEMENT HINT [ [ NO ] OVERRIDE ] [ COMMENT <comment> ] <statement_hint_target> 
  | REMOVE STATEMENT HINT [ ALL ] <statement_hint_target> }
```

**Syntax Elements**

**ADD STATEMENT HINT**

Adds hints specified in the `<hint_list>` to the specified target query or statement hash as a key. Separate multiple hints on the `<hint_list>` by a comma.

Evicts all cached plans for the target SQL statements that do not have the pattern symbols. For SQL statements using the pattern symbols, existing cached plans must be evicted manually prior to adding a hint.

**REMOVE STATEMENT HINT**

Removes statement hints from the system that match the specified target query or statement hash. If there are system hints, and the ALL option is not specified, then instead of removing the statement hints from the system, REMOVE STATEMENT HINT replaces the current user hints with the system hints. If the ALL option is specified, the statement removes the user hints and the system hints together with the specified SQL statement string. If ALL and `<statement_hash>` are specified, then user hints, system hints and the statement hash are removed. If `<statement_hash>` is specified without ALL, then the current user hints are replaced with the system hints.

Evicts all cached plans for the target SQL statements that do not have the pattern symbols. For SQL statements using the pattern symbols, existing cached plans must be evicted manually.

**hint_list**

Specifies a set of hints that are applied to the target query or statement_hash.

```
<hint_list> ::= <hint> [,... ]
```

**statement_hint_target**

Specifies the target of the hints to apply.

```
<statement_hint_target> ::= ...
```
ON { PROCEDURE | FUNCTION } proc_func_name

If a statement is specified as the target with the corresponding procedure/function name, the given hint is implicitly applied to the statement inside the corresponding procedure/function.

<proc_func_name> ::= [ <schema_name>.]<identifier>

target_query

Specifies the target query.

statement_hash

Specifies the statement hash.

[ NO ] OVERRIDE

Specifies whether to use a hint override with <statement_hint_target>. When OVERRIDE is specified, the hints specified in <hint_list> override any hints specified in <statement_hint_target> (e.g., SELECT * FROM myView WITH HINT (NO_CS_JOIN).

COMMENT comment

Specifies a comment to add to the STATEMENT_HINTS system view when <hint_list> is applied to <statement_hint_target>.

ALL

Removes all hints (user-defined and system from the system that are associated with the specified <statement_hint_target>). If ALL is not specified, then only the user-defined hints are removed. If ALL and <statement_hash> are specified, then user hints, system hints and the statement hash are removed. If <statement_hash> is specified without ALL, then the current user hints are replaced with the system hints.

Description

In a statement hint entry, there are two kinds of hints: system and user. User hints are the hints added by users with the ADD STATEMENT HINT defined above. System hints are the hints provided by the system by default when the system is upgraded.

In the case of conflicts, for example if user hints are already specified in the system for the SQL statement to which the system hints will be applied during the system upgrade, then the user hints stay in the system and the system hints are kept for later usage.

In the case of <hints_list> conflicts, for example USE_OLAPPLAN and NO_USE_OLAPPLAN, all hints are added to the target query without precedence or conflict resolution. This may cause queries to incur undefined behavior. Behavior is dependent on the implementation of each hint, making the result of the query containing the conflicts undeterminable.

ADD STATEMENT HINT adds hints specified in the <hints_list> to the specified target query or statement hash as a key. Separate multiple hints on the <hints_list> by a comma.

REMOVE STATEMENT HINT removes statement hints from the system that match the specified target query or statement hash. If there are system hints, and the ALL option is not specified, then instead of removing the statement hints from the system, REMOVE STATEMENT HINT replaces the current user hints with the system hints. If the ALL option is specified, the statement removes the user hints and the system hints together with the specified SQL statement string. If ALL and <statement_hash> are specified, then user hints, system hints
and the statement hash are removed. If `<statement_hash>` is specified without ALL, then the current user hints are replaced with the system hints.

Both commands (ADD and REMOVE) also evict all cached plans for the target SQL statements from the SQL Plan Cache.

**Example**

This example adds the NO_CS_JOIN and NO_CS_UNION_ALL hints to the SELECT query.

```
ALTER SYSTEM ADD STATEMENT HINT (NO_CS_JOIN, NO_CS_UNION_ALL) FOR SELECT * FROM AAA, BBB WHERE AAA.X = BBB.X;
```

This example alters the user hints associated with the specified SELECT statement, and specifies a comment.

```
ALTER SYSTEM ALTER STATEMENT HINT COMMENT 'AAA related join' FOR SELECT * FROM AAA, BBB WHERE AAA.X = BBB.X;
```

This example specifies an overriding hint to use with the specified SELECT statement.

```
ALTER SYSTEM ALTER STATEMENT HINT COMMENT 'AAA related join' FOR SELECT * FROM AAA, BBB WHERE AAA.X = BBB.X WITH HINT (NO_CS_JOIN);
```

This example removes all user hints associated with the specified SELECT statement.

```
ALTER SYSTEM REMOVE STATEMENT HINT FOR SELECT * FROM AAA, BBB WHERE AAA.X = BBB.X;
```

This example adds the NO_CS_JOIN hint to the statement hash 36896ef08346b8321f88449d5c5e97f8.

```
ALTER SYSTEM ADD STATEMENT HINT (NO_CS_JOIN) FOR STATEMENT HASH '36896ef08346b8321f88449d5c5e97f8';
```

This example removes all user hints associated with the statement hash 36896ef08346b8321f88449d5c5e97f8.

```
ALTER SYSTEM REMOVE STATEMENT HINT FOR STATEMENT HASH '36896ef08346b8321f88449d5c5e97f8';
```

This example removes all user and system hints and the statement hash 36896ef08346b8321f88449d5c5e97f8.

```
ALTER SYSTEM REMOVE STATEMENT HINT ALL FOR STATEMENT HASH '36896ef08346b8321f88449d5c5e97f8';
```

This example adds the NO_CS_JOIN, NO_CS_UNION_ALL hints on the MY_PROC procedure using the target query SELECT * FROM :TVAR AAA, BBB WHERE AAA.X = BBB.X.

```
ALTER SYSTEM ADD STATEMENT HINT (NO_CS_JOIN, NO_CS_UNION_ALL) ON PROCEDURE MY_PROC FOR SELECT * FROM :TVAR AAA, BBB WHERE AAA.X = BBB.X;
```
4.10.1.21.2 ALTER SYSTEM {ADD | ACTIVATE | UPDATE | DROP} KEY MANAGEMENT CONFIGURATION Statement (System Management)

Add, activate, update and drop key management configuration settings.

**Syntax**

```
ALTER SYSTEM ADD KEY MANAGEMENT CONFIGURATION <config_name> PROPERTIES '<settings>'
| ACTIVATE KEY MANAGEMENT CONFIGURATION <config_name>
| ALTER KEY MANAGEMENT CONFIGURATION <config_name> PROPERTIES '<updates>'
| DROP KEY MANAGEMENT CONFIGURATION <config_name>
```

**config_name** Specifies a unique name consisting of uppercase letters only, for example, 'AWS_HSM'.

**i Note**

The name 'DEFAULT' is activated by default. It becomes inactivated if you activate some other key management configuration (for example, to connect to DC KMS). Reactivate 'DEFAULT' to return to its default setting.

**settings**

Creates a JSON document with key-value settings. The list of keys and their supported values depends on the chosen KMS or HSM. The command will fail if the specified settings do not work.

**updates**

Modifies a JSON document with the new key-value settings. The list of keys and their supported values depends on the chosen HSM or KMS. All settings not described in the update remain unchanged.

**Permissions**

The use of this statement requires the DATABASE ADMIN and ENCRYPTION ROOT KEY ADMIN privileges.
Description

This can be used to switch from the current local secure store (LSS) configuration to an external key management service (KMS) or hardware security module (HSM).

Related Information

Using the LSS with an External Key Management System
Create and Manage a Key Management Configuration

4.10.1.21.3 ALTER SYSTEM {ADD | REMOVE} ABSTRACT SQL PLAN FILTER (System Management)

Starts or stops capturing abstract SQL plans for queries that match the specified filters.

Syntax

```
ALTER SYSTEM ADD ABSTRACT SQL PLAN FILTER <filter_name>
  SET <predicate_list>
| ALTER SYSTEM REMOVE ABSTRACT SQL PLAN FILTER <target_filter>
```

Syntax Element

filter_name

Specifies the name of the filter.

```
<filter_name>::= <string_literal>
```

SET predicate_list

Specifies the predicates that must be matched for the filter to be applied.

```
<predicate_list>::= <predicate>[, <predicate>...] ...
<predicate>::= '<key' = '<value>'
<key>::= APPLICATION USER NAME
    | APPLICATION NAME
    | USER NAME
    | SCHEMA NAME
    | XS APPLICATION USER NAME
<value>::= <string_literal>
```

REMOVE
Removes the specified abstract SQL plan filter.

**target_filter**

Specifies the name of a specific filter to remove or that all filters should be removed.

\[
<\text{target\_filter}> ::= \text{<filter\_name>} | \text{ALL}
\]

**Description**

Abstract SQL plans are captured for queries that all of the specified predicates for a filter. If there are multiple filters defined, then a query is checked against all filters, and if the query matches any filter, the query is captured.

**Example**

The following example defines a filter that captures abstract SQL plans for applications named App1.

```
ALTER SYSTEM ADD ABSTRACT SQL PLAN FILTER 'abc' SET 'APPLICATION NAME'='App1';
```

The following example defines a filter that captures abstract SQL plans for applications named App1 or with the application user name TEST.

```
ALTER SYSTEM ADD ABSTRACT SQL PLAN FILTER 'def' SET 'APPLICATION NAME'='App2', 'APPLICATION USER NAME'='TEST';
```

The following example deletes the filter named abc.

```
ALTER SYSTEM REMOVE ABSTRACT SQL PLAN FILTER 'abc';
```

The following example deletes all filters that have been defined.

```
ALTER SYSTEM REMOVE ABSTRACT SQL PLAN FILTER ALL;
```

**Related Information**

- SQL Plan Stability
- ALTER SYSTEM (ADD | REMOVE) ABSTRACT SQL PLAN FILTER (System Management) [page 497]
- ALTER SYSTEM (ENABLE | DISABLE | REMOVE) ABSTRACT SQL PLAN (System Management) [page 532]
- ALTER SYSTEM MIGRATE ABSTRACT SQL PLAN (System Management) [page 544]
- ALTER SYSTEM (START | STOP) APPLY ABSTRACT SQL PLAN (System Management) [page 570]
- ALTER SYSTEM UPDATE ABSTRACT SQL PLAN Statement (System Management) [page 586]
- ABSTRACT_SQL_PLAN System View [page 1477]
- M_ABSTRACT_SQL_PLAN_OVERVIEW System View [page 1730]
4.10.1.21.4 ALTER SYSTEM ALTER AUDIT LOG (System Management)

Change the internal table of the audit log to use SAP HANA Native Storage Extension (NSE).

Syntax

```
ALTER SYSTEM ALTER AUDIT LOG <load_unit>
```

Syntax Element

```
<load_unit> ::= [ {PAGE | DEFAULT | COLUMN } ] LOADABLE
```

For database versions 2.0 SPS 07 and later, by default, the audit log table is configured with the PAGE LOADABLE value. For prior database versions, by default, the audit log table is configured with the DEFAULT LOADABLE value.

Description

When `<load_unit>` is configured to PAGE LOADABLE, SAP HANA NSE only loads into memory those pages of the audit log table that are relevant to the queries on audit log. When configured to DEFAULT LOADABLE, the entire audit log table is loaded into memory.

Execution of the statement is audited as a mandatory audit event.

To determine the current value for `<load_unit>`, execute:

```
SELECT SCHEMA_NAME, TABLE_NAME, LOAD_UNIT FROM SYS.TABLES WHERE SCHEMA_NAME = '_SYS_AUDIT' AND TABLE_NAME = 'CS_AUDIT_LOG'
```
Privileges

Requires the AUDIT ADMIN system privilege.

Related Information

SAP HANA Native Storage Extension

4.10.1.21.5 ALTER SYSTEM ALTER CONFIGURATION Statement (System Management)

Sets or removes configuration parameters in an INI file.

Syntax

```
ALTER SYSTEM ALTER CONFIGURATION ( <filename>, <layer>[, <layer_name> ] )   { SET | UNSET } <parameter_key_value_list>   
[ WITH RECONFIGURE ]  
[ COMMENT <comment_string> ]
```

Syntax Elements

**filename**

The filename of the configuration file to be modified. If the file does not exist on the required layer, the file is created when a SET command is used.

```
<filename> ::= <string_literal>
```

**layer**

Sets the target layer for the configuration change. This parameter can be 'SYSTEM', 'HOST' or 'DATABASE'. The SYSTEM layer is the recommended layer for customer settings. The HOST layer should generally only be used for minor configuration, for example parameters contained in daemon.ini. In multitenant systems, system configuration files have an additional layer DATABASE to facilitate the configuration of properties for individual databases.

```
[layer] ::= '<HOST' | 'SYSTEM' | 'DATABASE'
```

**layer_name**
If the layer parameter above is set to 'HOST', `<layer_name>` is used to target either a tenant name or a target host name. For example, 'selxeon12' would target the 'selxeon12' host.

```plaintext
<layer_name> ::= <string_literal>
```

**i Note**
The 'HOST' value must be provided in lowercase only.

**SET**
Updates the value of a key if the key already exists, or inserts a new key if required.

**UNSET**
Removes a key and its associated value.

**parameter_key_value_list**
A list of configuration file entries to be modified or removed.

```plaintext
<parameter_key_value_list> ::= <parameter_key_value_entry> [{, <parameter_key_value_entry>}...]
```

**parameter_key_value_entry**
Specifies the section, key and value of the `.ini` file parameter to be created, modified, or removed.

```plaintext
<parameter_key_value_entry> ::= (<section_name>,<parameter_name>) [ = <parameter_value>]
```

**section_name**
Specifies the section name of the parameter to be modified

```plaintext
<section_name> ::= <string_literal>
```

**parameter_name**
Specifies the name of the parameter to be modified.

```plaintext
<parameter_name> ::= <string_literal>
```

**parameter_value**
Specifies the value of the parameter.

```plaintext
<parameter_value> ::= <string_literal>
```

**WITH RECONFIGURE**
Specifies that the configuration changes are directly applied to the running SAP HANA database instance.

When WITH RECONFIGURE is not specified the configuration changes are written to the required `.ini` file, however the modified values are not applied to the current running system. The changes are only applied during a restart of the SAP HANA database or a subsequent configuration change with WITH RECONFIGURE. In this case there can be inconsistencies between the `.ini` file contents and the actual configuration value that the SAP HANA database is currently using.

**COMMENT comment_string**
Specifies a free-text comment (string literal) that could be used to explain why an `.ini` file parameter is being changed.
Description

Sets or removes configuration parameters in an INI file. INI file configuration is used for the layered configuration of DEFAULT, SYSTEM, HOST layers. The DEFAULT layer configuration parameters cannot be changed or removed using this command.

By default, an error is returned if a settings is set to an unsupported value. You can control this behavior by using the unsupported_configuration_change parameter in the global.ini file.

The following is an example of .ini file locations:

- DEFAULT: /usr/sap/<SYSTEMNAME>/HDB<INSTANCENUMBER>/exe/config/indexserver.ini
- SYSTEM: /usr/sap/<SYSTEMNAME>/SYS/global/hdb/custom/config/indexserver.ini
- HOST: /usr/sap/<SYSTEMNAME>/HDB<INSTANCENUMBER>/<HOSTNAME>/indexserver.ini

The priority of the configuration layers is as follows: DEFAULT < SYSTEM < HOST.

Database-specific configuration files are located in the directory /hana/shared/$SID/global/hdb/custom/config/DB_<dbname>.

The layer that has the highest priority is the HOST layer, followed by the SYSTEM layer and finally the DEFAULT layer. The configuration with the highest priority is applied to the running environment. If the highest priority level configuration is removed, then the configuration with the next highest priority is applied.

For a list of configuration parameters, see the SAP HANA Configuration Parameter Reference.

Example

You set a parameter new_test_value in the alt_sys_test section of the global.ini file.

```
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'SYSTEM')
    SET ('alt_sys_test', 'new_test_value') = 'test';
```

You set the memory manager’s allocation limit setting to 50000 and provide a comment regarding the change:

```
ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'DATABASE', 'C11')
    SET ('memorymanager', 'allocationlimit') = '500000'
    WITH RECONFIGURE
    COMMENT 'Reverting to previous setting';
```

Related Information

SAP HANA Configuration Parameter Reference
M_INIFILES System View [page 1941]
M_INIFILE_CONTENT_HISTORY System View [page 1943]
M_INIFILE_CONTENTS System View [page 1942]
4.10.1.21.6 ALTER SYSTEM ALTER DATAVOLUME ADD PARTITION Statement (System Management)

Adds a new data volume partition to all index servers in the current tenant database or to a secondary system replication site. This statement must be executed on the indexserver, not on SystemDB.

Syntax

ALTER SYSTEM ALTER DATAVOLUME ADD PARTITION [ <path> ] [ SYSTEM REPLICAATION SITE <site> ]

Syntax Elements

path

Specifies a location for the new data volume. The path must be accessible from all nodes and services that reference the data volume.

<path> ::= PATH '/<directory>[/<subdirectory>[/…]]'

When this option is specified, the database server creates the folder structure mnt<xxxxx>/hdb<yyyyy>.<zzzzz> beneath the specified <path>.

Use of the <path> option must be enabled by setting the PERSISTENCE_DATAVOLUME_PARTITION_MULTIPATH setting in global.ini to TRUE.

site

Specifies the ID or name of the secondary site for the new data volume partition.

<site> ::= { <site_id> | <site_name> }
<site_id> ::= <unsigned_integer>
<site_name> ::= <string_literal>

Description

Use this command to add a new data volume partition to all index servers in the current tenant database or to a secondary system replication site.

When an ALTER SYSTEM ALTER DATAVOLUME ADD PARTITION statement is executed without specifying the PATH option, the SAP HANA server performs the following actions:

1. reserves a new partition on all index servers in the topology. Each reservation returns the reserved partition ID on that index server. If more than one partition ID is returned, the command fails.
2. adds information about the new partition to the indexserver.ini file.
activates the reserved partitions on all index servers in the topology. Activation is complete after the next 
save point on each index server. Once the new partition is active, new or changed data is distributed over it, 
as well as over all active data volume partitions.

Data volume partition IDs are stored in the M_DATA_VOLUME_PARTITION_STATISTICS system view.

When a new datavolume partition is added, it is inactive (M) and is activated on next savepoint.

Example

In the example below, a new data volume partition is added:

```
ALTER SYSTEM ALTER DATAVOLUME ADD PARTITION;
```

In the example below, a new data volume partition is added in the specified location:

```
ALTER SYSTEM ALTER DATAVOLUME ADD PARTITION PATH '/main/datavolumes';
```

Related Information

ALTER SYSTEM ALTER DATAVOLUME DROP PARTITION Statement (System Management) [page 504]
M_DATA_VOLUME_PARTITION_STATISTICS System View [page 1858]

4.10.1.21.7 ALTER SYSTEM ALTER DATAVOLUME DROP PARTITION Statement (System Management)

Drops a data volume partition from all index servers in the topology or from a secondary system replication 
site.

Syntax

```
ALTER SYSTEM ALTER DATAVOLUME DROP PARTITION <partition_id> [ SYSTEM REPLICATION 
SITE <site> ]
```

Syntax Elements

- **partition_id**
  - The ID of the data volume partition to drop.
site

Specifies the secondary site ID or name to drop the data volume partition from.

\[ <\text{site}> ::= \{ <\text{site\_id}> | <\text{site\_name}> \} \]
\[ <\text{site\_id}> ::= <\text{unsigned\_integer}> \]
\[ <\text{site\_name}> ::= <\text{string\_literal}> \]

**Description**

Use this command to drop a data volume partition from all index servers in the current topology.

Data volume partition IDs are stored in the M_DATA_VOLUME_PARTITION_STATISTICS system view.

When an ALTER SYSTEM ALTER DATAVOLUME DROP PARTITION statement is executed, the SAP HANA server:

1. moves all data that is stored in the partition to be dropped over to the remaining partitions. This operation can take considerable time depending on the size of the partition to be dropped.
2. removes information about the partition to be dropped from the `indexserver.ini` file.
3. removes the empty partition on each index server in the topology.

When system replication is running, you cannot drop an active data volume partition. To drop the partition, disable system replication functionality on the system, drop the partition, and re-enable system replication again.

**Example**

In the example below, data volume partition 1 is dropped:

```
ALTER SYSTEM ALTER DATAVOLUME DROP PARTITION 1;
```

**Related Information**

ALTER SYSTEM ALTER DATAVOLUME ADD PARTITION Statement (System Management) [page 503]
M_DATA_VOLUME_PARTITION_STATISTICS System View [page 1858]
4.10.1.21.8 ALTER SYSTEM ALTER SESSION SET Statement
(System Management)

Sets session variables for database sessions.

Syntax

ALTER SYSTEM ALTER SESSION <session_id> SET <key> = <value>

Syntax Elements

**session_id**
Specifies the session ID of the session where the variable should be set.

```plaintext
<session_id> ::= <unsigned_integer>
```

**key**
Specifies the key of a session variable.

```plaintext
<key> ::= <string_literal>
```

The maximum length of `key` is 32 characters.

**value**
Specifies the desired value of a session variable.

```plaintext
<value> ::= <string_literal>
```

The maximum length of `value` is 512 characters.

Description

For a list of predefined variables, see the `M_SESSION_CONTEXT` system view.

Session variables can be retrieved using the `SESSION_CONTEXT` command or from the `M_SESSION_CONTEXT` system view, and unset by using the ALTER SYSTEM ALTER SESSION UNSET statement.
Example

Run each command in this example in the same SQL editor you use. The same database session is required to make the example work properly.

Obtain your current session ID.

```sql
SELECT connection_id FROM m_connections WHERE OWN = 'TRUE';
```

Set the variable MY_VAR to 'some_value' in your session. In the command below, replace `<your_session_id>` with the session ID from the previous query.

```sql
ALTER SYSTEM ALTER SESSION <your_session_id> SET 'MY_VAR'= 'some_value';
```

Check the current value of MY_VAR in your session.

```sql
SELECT SESSION_CONTEXT('MY_VAR') FROM dummy;
```

4.10.1.21.9 ALTER SYSTEM ALTER SESSION UNSET

Statement (System Management)

Unsets session variables for database sessions.

Syntax

```sql
ALTER SYSTEM ALTER SESSION <session_id> UNSET <key>
```

Syntax Elements

**session_id**

Specifies the session ID of the session where the variable should be unset.

```sql
<session_id> ::= <unsigned_integer>
```

**key**

Specifies the key of a session variable.

```sql
<key> ::= <string_literal>
```

The maximum length of key is 32 characters.
Description

For a list of predefined variables, see the M_SESSION_CONTEXT system view.

Session variables can be retrieved using the SESSION_CONTEXT command or from the M_SESSION_CONTEXT system view, and unset by using the ALTER SYSTEM ALTER SESSION UNSET statement.

Example

Set the session variable MY_VAR to 'abc' in your database session.

```sql
SET 'MY_VAR' = 'abc';
```

Execute the following query to get a list of all the session variables of the current session.

```sql
SELECT * FROM M_SESSION_CONTEXT WHERE CONNECTION_ID = CURRENT_CONNECTION;
```

Remove the session variable from the specified session. In the statement below, replace `<your_session_id>` with the session ID that the previous statement returned.

```sql
ALTER SYSTEM ALTER SESSION <your_session_id> UNSET 'MY_VAR';
```

You get a list of all the session variables of the current session.

```sql
SELECT * FROM M_SESSION_CONTEXT WHERE CONNECTION_ID = CURRENT_CONNECTION;
```

From the results of this statement you can see that the MY_VAR variable has been unset.

4.10.1.21.10  ALTER SYSTEM ALTER TABLE PLACEMENT Statement (System Management)

Changes table classification and placement settings for table groups.

Syntax

```sql
ALTER SYSTEM ALTER TABLE PLACEMENT [ ( ( <table_classification_settings> ) ) ] [ SET | UNSET ] [ ( ( <table_placement_settings> ) ) ] ALTER SYSTEM ALTER TABLE PLACEMENT LOCATION <custom_location_specification>
```
Syntax Elements

table_classification_settings

Specifies one or more table group options.

```plaintext
<table_classification_settings> ::= <classification_key_value_pair> [,<...>]
<classification_key_value_pair> ::=    ... ]   | GROUP_NAME [=> <string_literal> ]   | GROUP_TYPE [=> <string_literal> ]   | SUB_TYPE [=> <string_literal> ]
```

**SCHEMA_NAME**

The schema name associated with the table (for example, SAPBWP).

**TABLE_NAME**

The table name (for example, /BIC/AZFIGL00).

**GROUP_NAME**

The table group name (for example, ZFIGL).

**GROUP_TYPE**

The table group type (for example, sap.bw.cube).

**SUB_TYPE**

The table group subtype. Possible values are ACTIVE, QUEUE, or CHANGE_LOG.

table_placement_settings

Specifies one or more table group placement properties.

```plaintext
<table_placement_settings> ::= <placement_key_value_pair> [...]
<placement_key_value_pair> ::=    ... ]   | LOCATION [ => <string_literal> ]   | SAME_PARTITION_COUNT [ => <boolean> ]
```

**MIN_ROWS_FOR_PARTITIONING [ => unsigned_integer]**

Defines the minimum number of records that must exist in a table before level 1 partitioning takes place (for example, 40,000).

**INITIAL_PARTITIONS [ => unsigned_integer]**

Defines the initial number of partitions. If the threshold value in the MIN_ROWS_FOR_PARTITIONING column is exceeded, the table redistribution performs a partitioning (for example, 3).

**REPARTITIONING_THRESHOLD [ => unsigned_integer]**

Specifies the threshold value for the number of records in a partition that triggers a repartitioning (for example, 2,000,000,000).

Once a table has been partitioned with the specified initial number of partitions, for performance reasons, the table is only repartitioned by doubling the number of partitions. For example, if the initial number of partitions is 3, this would result in 6 partitions being created during a repartitioning.

**LOCATION [ => string_literal]**
Specifies one or more indexserver volume IDs (values are MASTER, DEFAULT, SLAVE, and ALL), or the name of a custom set of indexservers previously defined using the ALTER TABLE ALTER TABLE PLACEMENT LOCATION syntax.

**SAME_PARTITION_COUNT [ => boolean]**

Specifies that all partitions of the tables in a group will contain the same number of partitions.

**LOCATION custom_location_specification**

Defines a synonym for a list of one or more SAP HANA indexservers, or groups of indexservers, by their intended use; for example, as a worker for a specific application to isolate the workload, or to define a custom subset of indexservers that host data of a specific customer. This capability relies more on volume IDs instead of host/port names.

```
LOCATION <custom_location_specification> ::= <custom_location_names> { SET <location_specification> | UNSET }
```

**custom_location_names**

Specifies a synonym for one or more volume IDs to use for table placement. You can then reference this name when specifying table placement settings using the ALTER TABLE ALTER TABLE PLACEMENT syntax.

**SET location_specification**

Specifies the locations to include or exclude during table placement.

```
<location_specification> ::= { [ <include_list> [, <exclude_list> ] ] }  
<include_list> ::= INCLUDE=>'[ { <preset_value> [, ...] | <volumn_id> [, ...] } ]'  
<exclude_list> ::= EXCLUDE=>'[ { <preset_value> [, ...] | <volumn_id> [, ...] } ]'  
<preset_value> ::= { MASTER | ALL | DEFAULT | SLAVE }
```

**UNSET**

Unsets the locations configured for the specified <custom_location_specification>.

If <custom_location_specification> is UNSET while also being the table placement setting currently in use, SAP HANA reverts to default table placement behaviors.

**Description**

This statement allows you to set and unset attributes and placement properties for table groups. When you create a table, you can assign it to a table group. Table groups can improve the database server's ability to make decisions about how to distribute tables across servers. For example, the use of table groups can prevent frequently-joined tables from being placed on different hosts.

Table classification and the table placement settings are taken into account on two occasions: table redistribution and during table creation.

Table placement attributes and properties are stored in the TABLE_GROUPS and TABLEPlacement LOCATIONS system views and their corresponding monitoring view counterparts.
Permissions

You must have the TABLE ADMIN to execute this statement.

Examples

The following statement forces all tables to be stored on the slave host.

```
ALTER SYSTEM ALTER TABLE PLACEMENT SET (LOCATION => 'slave');
```

The following statement forces all tables of the schema SAPKIT to be stored on the master host.

```
ALTER SYSTEM ALTER TABLE PLACEMENT (SCHEMA_NAME => 'SAPKIT') SET (LOCATION => 'master');
```

The following statement forces all tables of the schema SAPKIT and the GROUP_TYPE sap bw dso to be stored on the slave host with the specified properties.

```
ALTER SYSTEM ALTER TABLE PLACEMENT (SCHEMA_NAME => 'SAPKIT', GROUP_TYPE => 'sap bw dso')    SET (LOCATION => 'slave', MIN_ROWS_FOR_PARTITIONING => 40000000, REPARTITIONING_THRESHOLD => 40000000, INITIAL_PARTITIONS => 3);
```

The following example demonstrates how to unset the schema entry SAPKIT and GROUP_TYPE sap bw dso.

```
ALTER SYSTEM ALTER TABLE PLACEMENT (SCHEMA_NAME => 'SAPKIT', GROUP_TYPE => 'sap bw dso') UNSET;
```

The following example demonstrates how to remove the LOCATION entry for schema SAPKIT and GROUP_TYPE sap bw dso.

```
ALTER SYSTEM ALTER TABLE PLACEMENT (SCHEMA_NAME => 'SAPKIT', GROUP_TYPE => 'sap bw dso') UNSET (LOCATION);
```

The following example demonstrates how to set the SAME_PARTITION_COUNT table placement property for schema SYSTEM to TRUE.

```
ALTER SYSTEM ALTER TABLE PLACEMENT (SCHEMA_NAME=>'SYSTEM') SET (SAME_PARTITION_COUNT => 'TRUE');
```

The following example demonstrates how to unset the SAME_PARTITION_COUNT table placement property for schema SYSTEM.

```
ALTER SYSTEM ALTER TABLE PLACEMENT (SCHEMA_NAME=>'SYSTEM') UNSET (SAME_PARTITION_COUNT);
```

The following examples demonstrate how to include or exclude locations for table placement:

```
ALTER SYSTEM ALTER TABLE PLACEMENT LOCATION MyLocation SET (INCLUDE => '2,3', EXCLUDE => 'slave');
ALTER SYSTEM ALTER TABLE PLACEMENT LOCATION MyLocation SET (INCLUDE => '2,3', EXCLUDE => '');
```
The following example unsets the list of locations that have been set for table placement:

```sql
ALTER SYSTEM ALTER TABLE PLACEMENT LOCATION MyLocation UNSET;
```

**Related Information**

**Table Placement**
- TABLE_PLACEMENT System View [page 1684]
- M_EFFECTIVE_TABLE_PLACEMENT System View [page 1880]
- TABLE_PLACEMENT_LOCATIONS System View [page 1685]
- M_TABLE_PLACEMENT_LOCATIONS System View [page 2217]
- TABLE_GROUPS System View [page 1680]

### 4.10.1.21.11 ALTER SYSTEM APPLICATION ENCRYPTION Statement (System Management)

Manages encryption keys for applications that use the internal data encryption service.

**Syntax**

```sql
ALTER SYSTEM APPLICATION ENCRYPTION <encrypt_option>
```

**Syntax Elements**

**encrypt_option**

Specifies application encryption options.

```sql
<encrypt_option> ::= CREATE NEW KEY
                      | CREATE NEW ROOT KEY WITHOUT ACTIVATE
                      | ACTIVATE NEW ROOT KEY
```

**CREATE NEW KEY**

Creates a new, random encryption key for every application.

**CREATE NEW ROOT KEY WITHOUT ACTIVATE**

The CREATE NEW ROOT KEY WITHOUT ACTIVATE clause creates a new ACTIVE root key, re-encrypts all application keys using the new key, and records the new key in the redo log.
The new root key is in the PREACTIVE state. A PREACTIVE key is not recorded in the redo log and cannot be used to re-encrypt the application encryption keys. This clause returns an error if a PREACTIVE key already exists. Creating a PREACTIVE application root key allows you to back up the key before using it to encrypt application keys.

**ACTIVATE NEW ROOT KEY**

Activates a PREACTIVE key that was previously created by using the CREATE NEW ROOT KEY WITHOUT ACTIVATE clause. This clause updates the state of the PREACTIVE root key to ACTIVE and the previously ACTIVE root key to DEACTIVATED. An error is returned if a PREACTIVE root key does not exist.

This clause re-encrypts all application keys by using the activated root key.

**Description**

You can use this statement to manage random encryption keys for application encryption and create and activate root keys.

Applications are consumers of the internal data encryption service, such as internal components, like the secure internal credential store, or XS applications. The new keys are stored encrypted with the current root key of the internal data encryption service. The new keys are used after the transaction is committed. No changes are made to data that has already been written to disk.

Use this statement if your company’s security policies dictate that you should periodically change the encryption keys being used to store your data, or if you are instructed to do so by SAP Support.

You must back up the root key store between creating and activating a new root key. You can back up the root key store by using the ENCRYPTION_ROOT KEYS_EXTRACT KEYS function and saving the results in a backup file.

For more information about encryption in the SAP HANA database and the key change process, see the *SAP HANA Administration Guide* and the *SAP HANA Security Guide*.

**Permissions**

You must have the ENCRYPTION ROOT KEY ADMIN privilege to execute this statement.

You must have the ENCRYPTION ROOT KEY ADMIN or RESOURCE ADMIN privilege to create application encryption keys.

**Example**

Create new encryption keys for all applications.

```
ALTER SYSTEM APPLICATION ENCRYPTION CREATE NEW KEY;
```
Related Information

ENCRYPTION_ROOT_KEYS_EXTRACT KEYS Function (Security) [page 187]
Server-Side Data Encryption Services
Managing Data Encryption in SAP HANA
ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 568]
ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
ENCRYPTION_ROOT_KEYS_EXTRACT KEYS Function (Security) [page 187]
ENCRYPTION_ROOT KEYS System View [page 1558]
APPLICATION_ENCRYPTION KEYS System View [page 1498]
M_ENCRYPTION_OVERVIEW System View [page 1881]
ALTER SYSTEM LOG ENCRYPTION Statement (System Management) [page 541]

4.10.1.21.12 ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management)

Controls whether data and log backups are stored on disk in an encrypted or non-encrypted format and manages the encryption root keys used.

Syntax

Alter System Backup Encryption <encrypt_option>

Syntax Elements

encrypt_option

Specifies backup encryption options.

<encrypt_option> ::= CREATE NEW ROOT KEY WITHOUT ACTIVATE | ACTIVATE NEW ROOT KEY | ON | OFF

CREATE NEW ROOT KEY WITHOUT ACTIVATE

Creates a new random encryption root key for backup encryption with the key state PREACTIVE. Once activated, the new key is used to encrypt any new backups. You must backup the key before it is used to encrypt new backups.

Execute this syntax in the database for which the root key is being created.
An error occurs if a PREACTIVE key already exists and an ALTER SYSTEM BACKUP ENCRYPTION CREATE NEW ROOT KEY WITHOUT ACTIVATE statement is executed. Similarly, an error occurs when a PREACTIVE key does not exist and an ALTER SYSTEM BACKUP ENCRYPTION ACTIVATE NEW ROOT KEY statement is executed.

You must back up the root key store between creating and activating a new root key. You can back up the root key store using BACKUP ENCRYPTION ROOT KEYS or the functions ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS and ENCRYPTION_ROOT_KEYS_EXTRACT_ALL_KEYS_FOR_DATABASE function and saving the results to a backup file.

**ACTIVATE NEW ROOT KEY**

Updates the state of the PREACTIVE root key to ACTIVE, and the previously ACTIVE root key to DEACTIVATED. Once activated, the new key is used for encrypting any new backups.

**ON** Turns on backup encryption. Any new backups created after backup encryption is enabled are encrypted using the ACTIVE backup encryption root key.

**OFF** Turns off backup encryption. Any new backups created after backup encryption is turned off are written in cleartext. Existing backups remain encrypted.

### Description

You can use this statement to enable or disable backup encryption, and create and activate encryption root keys.

If there is no encryption configuration in the root key store (instance SSFS or LSS), then encryption is interpreted as OFF.

Execute this statement on the database where you want the encryption option to take effect.

**Note**

If the system database has encryption configuration control, data volume encryption can only be enabled or disabled in a tenant database from the system database using ALTER DATABASE <database_name> BACKUP ENCRYPTION ON.

For more information about encryption in the SAP HANA database and the root key change process, see the *SAP HANA Security Guide* and the *SAP HANA Administration Guide*.

### Permissions

You must have the ENCRYPTION ROOT KEY ADMIN privilege to execute this statement.
Example

Create a new backup root key in the preactive state:

```sql
ALTER SYSTEM BACKUP ENCRYPTION CREATE NEW ROOT KEY WITHOUT ACTIVATE;
```

Related Information

ALTER SYSTEM VALIDATE ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 587]
M_PERSISTENCE_ENCRYPTION_KEYS System View [page 2042]
ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS Function (Security) [page 187]
ALTER DATABASE Statement (Tenant Database Management) [page 440]
Backup Encryption
Managing Data Encryption in SAP HANA
ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 568]
ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS Function (Security) [page 187]
ENCRYPTION_ROOT_KEYS_EXTRACT_ALL_KEYS_FOR_DATABASE Function (Security) [page 188]
ENCRYPTION_ROOT_KEYS System View [page 1558]
APPLICATION_ENCRYPTION_KEYS System View [page 1498]
M_ENCRYPTION_OVERVIEW System View [page 1881]
ALTER SYSTEM LOG ENCRYPTION Statement (System Management) [page 541]
BACKUP ENCRYPTION ROOT KEYS Statement (Backup and Recovery) [page 709]
RECOVER ENCRYPTION ROOT KEYS Statement (Backup and Recovery) [page 1079]

4.10.1.21.13 ALTER SYSTEM CANCEL [WORK IN] SESSION Statement (System Management)

Cancels the currently executing statement of a session.

Syntax

```sql
ALTER SYSTEM CANCEL [WORK IN] SESSION <connection_id>
```
Syntax Elements

**connection_id**

Specifies the connection ID of the session.

\[<\text{connection_id}> ::= <\text{string}\_\text{literal}>\]

**Description**

The transaction of the canceled session is rolled back. The statement that was executing returns error code 139 (current operation canceled by request and transaction rolled back).

**Example**

The following query returns the current database connection IDs and the statements that the sessions are executing.

```
SELECT C.CONNECTION_ID, PS.STATEMENT_STRING
FROM M_CONNECTIONS C JOIN M_PREPARED_STATEMENTS PS
ON C.CONNECTION_ID = PS.CONNECTION_ID AND C.CURRENT_STATEMENT_ID = PS.STATEMENT_ID
WHERE C.CONNECTION_STATUS = 'RUNNING' AND C.CONNECTION_TYPE = 'Remote';
```

Using the connection ID information that you obtained using the query above, you can now cancel a running query. In the statement below, replace `<connection_id>` with a connection ID from the query above.

```
ALTER SYSTEM CANCEL SESSION '<connection_id>';
```

### 4.10.1.21.14 ALTER SYSTEM CLEAR AUDIT LOG Statement

**(System Management)**

Deletes old audit data from the SAP HANA database audit table.

**Syntax**

```
ALTER SYSTEM CLEAR AUDIT LOG [ FOR AUDIT POLICY <policy_name> ]
<until_specification>
```
Syntax Elements

until_specification
Removes audit data older than the <timestamp>.

\[
\begin{align*}
\text{until_specification} & \ ::= \ { \text{UNTIL } \text{<timestamp>} \ | \ \text{ALL} } \\
\text{<timestamp>} & \ ::= \ \text{<string_literal>}
\end{align*}
\]

If the ALL keyword is used, then all the audit data is removed.

Description

Use this command to delete old audit data from the SAP HANA database audit table.

Example

Delete audit log data older than December 31st 2012.

```
ALTER SYSTEM CLEAR AUDIT LOG UNTIL '2012-12-31 23:59:59';
```

Delete all audit log data for audit policy MY_POLICY.

```
ALTER SYSTEM CLEAR AUDIT LOG FOR AUDIT POLICY MY_POLICY ALL;
```

Related Information

AUDIT_LOG System View [page 1501]

4.10.1.21.15 ALTER SYSTEM CLEAR CACHE Statement (System Management)

Clears resources (entries) from one or more cache instances.

Syntax

```
ALTER SYSTEM CLEAR CACHE ( <cache_id> [, <cache_id> [...] ] [ SYNC ] )
```
Syntax Elements

**cache_id**
Specifies the ID of the cache instance to clear, as found in the M_CACHES system view.

**SYNC**
Increases the retry attempts for resources (entries) that were initially blocked from clearing. Specifying SYNC may make this statement take longer to return (for example if many retries were required) but performs a better clearing operation than without specifying SYNC.

Description

Executing this statement clears the specified cache instances not only on the local (index) server, but across all distributed instances of the cache.

Entries (resources) in a cache instance may be blocked from clearing. This can happen while entries are being inserted or searched for, or when entries are being fetched for display in M_CACHE_ENTRIES. There may be other reasons why entries are blocked from clearing as well, so after a clearing operation (even if SYNC is specified), some entries may still remain in the cache instance.

When SYNC is not specified, the HANA server makes a best attempt effort at clearing as many cache entries as possible, and the statement always succeeds.

When SYNC is specified, the HANA server increases the number of retry attempts on resources that were initially busy. If there are distributed instances of the cache, then the command is successful only if each local clearing operation succeeds without exceeding the timeout for cache distributed requests (ini parameter [cache] resultcache_request_timeout_inMilliseconds). With SYNC, the statement may return a general error (error number 2) if the clear operation failed to clear some entries.

Regardless of whether SYNC is specified, the M_CACHE_ENTRIES system view may still return entries immediately after the ALTER SYSTEM CLEAR CACHE statement is run. This can happen if the clearing operation occurred concurrently with other database operations.

Related Information

M_CACHE_ENTRIES System View [page 1768]
### 4.10.1.21.16 ALTER SYSTEM CLEAR COLUMN JOIN DATA STATISTICS (System Management)

Removes join data statistics from the SAP HANA database cache.

#### Syntax

```
ALTER SYSTEM CLEAR COLUMN JOIN DATA STATISTICS
```

#### Description

Clears all join data statistics from the system so that the next time a join execution is run join data statistics are recalculated in a deterministic way.

#### Examples

Retrieve, display, and delete column join data statistics.

```sql
DROP TABLE A;
CREATE COLUMN TABLE A (aa NVARCHAR(2), ab INT);
DROP TABLE B;
CREATE COLUMN TABLE B (b1 INT, b2 NVARCHAR(1));
INSERT INTO A (aa, ab) VALUES ('AA', 1);
INSERT INTO A (aa, ab) VALUES ('BB', 2);
INSERT INTO B (b1, b2) VALUES (1, 'a');
INSERT INTO B (b1, b2) VALUES (3, 'f');
MERGE DELTA OF A;
MERGE DELTA OF B;
```

Perform a join on tables A and B.

```sql
SELECT * FROM A INNER JOIN B ON A.ab = B.b1;
```

View the column join data statistics for tables A and B.

```sql
SELECT COUNT(*) FROM M_JOIN_DATA_STATISTICS WHERE schema_name1 = 'SYSTEM' AND table_name1 IN ('A', 'B');
SELECT * FROM M_JOIN_DATA_STATISTICS;
```

Clear the column join data statistics from the system.

```sql
ALTER SYSTEM CLEAR COLUMN JOIN DATA STATISTICS;
```

Clear the column join data statistics from table A.

```sql
ALTER TABLE A CLEAR COLUMN JOIN DATA STATISTICS;
```
4.10.1.21.17 ALTER SYSTEM CLEAR INIFILE CONTENT HISTORY Statement (System Management)

Clears ini file content history from the catalog.

Syntax

```
ALTER SYSTEM CLEAR INIFILE CONTENT HISTORY
```

Description

When you use this statement, the specified ini file catalog history data is cleared from the M_INIFILE_CONTENT_HISTORY system view and underlying table.

In addition to deleting the records of ini file changes, this statement also inserts a record into M_INIFILE_CONTENT_HISTORY to indicate the operation.

Permissions

You must have the MONITOR ADMIN privilege to execute this statement.

Related Information

- M_INIFILE_CONTENTS System View [page 1942]
- M_INIFILE_CONTENT_HISTORY System View [page 1943]
- M_INIFILES System View [page 1941]
4.10.1.21.18  ALTER SYSTEM CLEAR RESULT CACHE
Statement (System Management)

Removes all result cache entries from the system.

Syntax

```sql
ALTER SYSTEM CLEAR [<cache_type>] RESULT CACHE
```

Syntax Elements

cache_type

Setting `<cache_type>` to DYNAMIC ensures that the cache is refreshed every time a query on the result cache is run. The default value is STATIC.

```sql
<cache_type> ::= STATIC | DYNAMIC
```

Description

The next suitable SELECT statement rebuilds the respective result cache entry.

4.10.1.21.19  ALTER SYSTEM CLEAR SQL PLAN CACHE
Statement (System Management)

Removes all of the SQL plans that are not currently being executed from the SAP HANA database plan cache.

Syntax

```sql
ALTER SYSTEM CLEAR SQL PLAN CACHE
```
Description

The ALTER SYSTEM CLEAR SQL PLAN CACHE statement removes all plans with a reference count of 0 from the plan cache and resets all the statistics of the remaining plans. The command also resets the contents of M_SQL_PLAN_CACHE_OVERVIEW monitoring view.

The SQL PLAN CACHE stores plans generated by previous SQL statement executions. The SAP HANA database uses the plan cache to speed up query execution if the same SQL statement is executed multiple times. The plan cache also collects some statistics regarding plan preparation and execution.

Permissions

You must have the OPTIMIZER ADMIN privilege.

Example

Clear the SQL plan cache.

```
ALTER SYSTEM CLEAR SQL PLAN CACHE;
```

Related Information

M_SQL_PLAN_CACHE System View [page 2145]
M_SQL_PLAN_CACHE_OVERVIEW System View [page 2157]

4.10.1.21.20 ALTER SYSTEM CLEAR TIMEZONE CACHE Statement (System Management)

Clears cached timezone definitions.

Syntax

```
ALTER SYSTEM CLEAR TIMEZONE CACHE DATASET <timezone_dataset_string_literal>
```
Syntax Elements

timezone_dataset_string_literal

Specifies the data to be cleared.

<timezone_dataset_string_literal> ::= { sap | platform }

sap

Specifies to clear the currently-used SAP HANA dataset.

platform

Specifies to clear the dataset provided by the operating system.

Description

For efficiency purposes, timezone definitions are cached in internal structures for timestamp conversions such as UTC to local date and time. Use ALTER SYSTEM CLEAR TIMEZONE CACHE DATASET to clear out the cached timezone definitions when they become stale, for example when a timezone definition changes.

After executing this statement, the timezone cache contents are recreated for the respective data sources at the next access.

Permissions

The SERVICE ADMIN privilege is required to execute this statement.

Example

Clears all cached timezone definitions in the currently used 'sap' dataset.

ALTER SYSTEM CLEAR TIMEZONE CACHE DATASET 'sap';

Related Information

TIMEZONES System View [page 1692]
4.10.1.21.21 ALTER SYSTEM CLEAR TRACES Statement
(System Management)

Clears (removes) trace files opened by SAP HANA.

Syntax

```
ALTER SYSTEM CLEAR TRACES { <trace_type_list> } [ UNTIL <timestamp> ] [ WITH BACKUP ]
```

Syntax Elements

`trace_type_list`
Specifies a list of trace types to be cleared.

```
<trace_type_list> ::= <trace_type> [, <trace_type> [,... ]]
```

`trace_type`
Specifies the trace type.

```
<trace_type> ::= <string_literal>
```

You can selectively clear specific trace files by setting `<trace_type>` to one of the following types:

<table>
<thead>
<tr>
<th><code>&lt;trace_type&gt;</code></th>
<th>Trace Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>all *.trc files of services listed below</td>
</tr>
<tr>
<td>ALERT</td>
<td><em>alert_</em>.trc</td>
</tr>
<tr>
<td>BACKUP</td>
<td>backup.log</td>
</tr>
<tr>
<td>BACKINT</td>
<td>backint.log</td>
</tr>
<tr>
<td>CLIENT</td>
<td>localclient_*.trc</td>
</tr>
<tr>
<td>CRASHDUMP</td>
<td><em>crashdump.</em></td>
</tr>
<tr>
<td>EMERGENCYDUMP</td>
<td><em>emergencydump.</em></td>
</tr>
<tr>
<td>EXPENSIVESTATEMENT</td>
<td><em>expensive_statements.</em>.trc</td>
</tr>
<tr>
<td>indexserver,nameserver,...,daemon</td>
<td>open *.trc files of a single service type</td>
</tr>
</tbody>
</table>
trace_type

<table>
<thead>
<tr>
<th>Trace Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROWSTOREREORG <em>.row_store_reorg.</em>.trc</td>
</tr>
<tr>
<td>RTEDUMP <em>.rtedump.</em>.trc</td>
</tr>
<tr>
<td>SQLTRACE sqltrace*.py</td>
</tr>
<tr>
<td>TABLE PARTITION OPERATIONS indexserver_<em>.table_partition_operation.</em>.trc</td>
</tr>
<tr>
<td>UNLOAD <em>.unloads.</em>.trc</td>
</tr>
</tbody>
</table>

timestamp

Specifies that all trace files with a modification time before or equal to the timestamp are deleted.

<timestamp> ::= <string_literal>

WITH BACKUP Specifies that trace files are compressed and saved instead of removed.

Description

When you use this statement, all trace files and compressed files (like .gz) that were opened by the SAP HANA database are removed, assuming WITH BACKUP is not specified.

On distributed systems, this command clears all trace files on all hosts.

Use this command to reduce disk space used by large trace files for example, when trace components are set to INFO or DEBUG.

Examples

Clear the alert trace file.

```
ALTER SYSTEM CLEAR TRACES ('ALERT');
```

Clear the alert and client trace files.

```
ALTER SYSTEM CLEAR TRACES ('ALERT', 'CLIENT');
```

Back up the alert and client trace files.

```
ALTER SYSTEM CLEAR TRACES ('ALERT', 'CLIENT') WITH BACKUP;
```

Clear the alert and client trace files with a timestamp prior or equal to 2015-12-31 23:59:59.999.

```
ALTER SYSTEM CLEAR TRACES ('ALERT', 'CLIENT') UNTIL '2015-12-31 23:59:59';
```
Back up the indexserver trace files.

```
ALTER SYSTEM CLEAR TRACES('indexserver') WITH BACKUP;
```

**Related Information**

- ALTER SYSTEM REMOVE TRACES Statement (System Management) [page 564]
- M_TRACEFILES System View [page 2246]
- ALTER SYSTEM [START | STOP] PERFTRACE Statement (System Management) [page 578]
- M_TRACEFILES System View [page 2246]
- M_TRACEFILE_CONTENTS System View [page 2247]

### 4.10.1.21.22 ALTER SYSTEM CLIENTPKI \{ UPDATE / DROP \} (System Management)

Manage CLIENTPKI certificates and authorities.

**Syntax**

```
ALTER SYSTEM CLIENTPKI UPDATE \{ CERTIFICATES | ROOT CA \}
 | ALTER SYSTEM CLIENTPKI DROP ROOT CA
```

**Syntax Elements**

- **UPDATE CERTIFICATES**
  Recreates the host certificate when SAP HANA database is running.

- **UPDATE ROOT CA**
  Recreates the root and host certificate when SAP HANA database is running

- **DROP ROOT CA**
  Removes the root certificate when SAP HANA database is running, if sslclientpki is disabled (OFF)

**Remarks**

Each time SAP HANA database is restarted, the host certificate is regenerated with 4096 RSA key. The root certificate is stored as a file for use in SAP HANA client as SSL trust.
For more information on TLS/SSL configuration, see TLS/SSL Configuration on the SAP HANA Server

Permissions

You must have the SSL ADMIN system privilege to use this statement.

4.10.1.21.23 ALTER SYSTEM CREATE RUNTIMEDUMP
Statement (System Management)

Creates a runtimedump (RTE) dump file containing information about specific sections and profiles.

Syntax

```
ALTER SYSTEM CREATE RUNTIMEDUMP [ AT [ LOCATION ] <location_ref> ]
[ [ { SECTION <section_ref> | PROFILE <profile_name> } ] ] [ INTO FILE
' <file_name> ' ]

<location_ref> ::= '<host>:<port>'

If no location is specified, then the trace folder of the specified service is used.

SECTION

Specifies one of the section names defined in the runtimedump section in the global.ini file.

<section_ref> ::= ( ' <section_name> ' | , ' <section_name> ' , ... )

<section_name> ::= <string_literal>

If no section is specified, then all sections are written.

PROFILE

Specifies the named subset of sections defined in the runtimedump section in the global.ini file.

INTO FILE

Specifies the name for the dump file. If a file name is not specified, then the default file name

<servicename>_<hostname>_<port>_<YYYYMMDD-HHMMSS>_<pid>_trc is used.
**Permissions**

You need the RESOURCE_ADMIN system privilege.

**Description**

This syntax lets you create an RTE dump file containing information about specific sections and profiles.

**Examples**

Writes information relating to the STACK_SHORT section only to a file named `my_rte_dump.trc` on myhost, port 30003.

```sql
ALTER SYSTEM CREATE RUNTIMEDUMP AT LOCATION 'myhost:30003' SECTIONS ('STACK_SHORT') INTO FILE 'my_rte_dump.trc';
```

Writes information relating to all sections defined by the profile named myRTEProfile on the connected service to the file named `my_rte_dump.trc`.

```sql
ALTER SYSTEM CREATE RUNTIMEDUMP PROFILE 'myRTEProfile' INTO FILE 'my_rte_dump.trc';
```

**4.10.1.21.24 ALTER SYSTEM CREATE WAITGRAPH Statement (System Management)**

Create a waitgraph file that contains thread information about deadlocks.

**Syntax**

```sql
ALTER SYSTEM CREATE WAITGRAPH [ AT [ LOCATION ] <location_ref> ] [ ALL THREADS ] [ INTO FILE <file_name> ]
```

**Syntax Elements**

- `location_ref`
Specifies the location of the service.

\[
<location_ref> ::= '<host>:<port>'
\]

If no location is specified, then the trace folder of the specified service is used.

**ALL THREADS** Includes details on all threads on the connected service, not just those that are part of the deadlock.

**INTO FILE**

Specifies the name for the dump file. If a filename is not specified, then the default file name
\[
<servicename>_<hostname>_<port>.waitgraph.<YYYYMMDD-HHMMSS>_<pid>.trc
\]
is used.

**Permissions**

You need the RESOURCE_ADMIN system privilege.

**Description**

This syntax lets you create a dump file containing information about threads that may participate in deadlocks.

**Examples**

Write the current waitgraph on service myhost:30003 to the file named my_waitgraph.trc.

```
ALTER SYSTEM CREATE WAITGRAPH AT LOCATION 'myhost:30003' INTO FILE 'my_waitgraph.trc';
```

Write the current waitgraph (including all threads even if they do not participate on the deadlock) on the connected service into the file named my_waitgraph.trc.

```
ALTER SYSTEM CREATE WAITGRAPH ALL THREADS INTO FILE 'my_waitgraph.trc';
```
4.10.1.21.25 ALTER SYSTEM DISCONNECT SESSION
Statement (System Management)

Disconnects the specified connection session from the database.

Syntax

```
ALTER SYSTEM DISCONNECT SESSION <session_id>
```

Syntax Elements

```
<session_id> ::= <string_literal>
```

Description

Before disconnecting, any currently running operations associated with the session are terminated.

If `<session_id>` is an internal connection, the statement is ignored.

Example

The following query obtains the session IDs of idle sessions.

```
SELECT CONNECTION_ID, IDLE_TIME FROM M_CONNECTIONS
   WHERE CONNECTION_STATUS = 'IDLE' AND CONNECTION_TYPE = 'Remote'
   ORDER BY IDLE_TIME DESC;
```

To disconnect a session, replace `<connection_id>` with a connection ID from the query above.

```
ALTER SYSTEM DISCONNECT SESSION '<connection_id>';```
4.10.1.21.26  ALTER SYSTEM {ENABLE | DISABLE | REMOVE} ABSTRACT SQL PLAN (System Management)

Enables or disables execution plan generation for abstract SQL plans, or removes plans from the ABSTRACT_SQL_PLANS table.

Syntax

```
ALTER SYSTEM <enable_disable_remove_action> ABSTRACT SQL PLAN ENTRY <target>
```

Syntax Elements

**enable_disable_remove_action**
Specifies the action to perform.

```
<enable_disable_remove_action> ::= { ENABLE | DISABLE | REMOVE }
```

**ENABLE**
Enables the abstract SQL plans for use by the optimizer.

**DISABLE**
Disables the abstract SQL plans for use by the optimizer.

**REMOVE**
Permanently deletes the abstract SQL plans.

**target**
Specifies the scope of `<action>`.

```
<target> ::= ALL | FOR <id_list>
/id_list> ::= <plan_id> [, <plan_id> [, , ... ]]
```

**id_list** Specifies the plans to which the action applies.

**<id_list>** applies the **<action>** as follows:

<table>
<thead>
<tr>
<th>Action</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE</td>
<td>Enable the abstract SQL plan entries specified in the <code>&lt;id_list&gt;</code>. If multiple abstract SQL plan entries of the same query are specified in the <code>&lt;id_list&gt;</code>, then the last entry of the same query in the list is enabled.</td>
</tr>
<tr>
<td>DISABLE</td>
<td>Disable the abstract SQL plan entries specified in the <code>&lt;id_list&gt;</code>.</td>
</tr>
<tr>
<td>Action</td>
<td>Behavior</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REMOVE</td>
<td>Remove the abstract SQL plan entries specified in the <code>&lt;id_list&gt;</code>.</td>
</tr>
</tbody>
</table>

**ALL**

ALL applies the action to all abstract SQL plans.

ALL applies the `<action>` to all abstract SQL plans, as follows:

<table>
<thead>
<tr>
<th>Action</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE</td>
<td>For each abstract SQL plan query that does not have any enabled entry, enable the first captured entry.</td>
</tr>
<tr>
<td>DISABLE</td>
<td>Disable all enabled abstract SQL plan entries.</td>
</tr>
<tr>
<td>REMOVE</td>
<td>Remove all abstract SQL plan entries.</td>
</tr>
</tbody>
</table>

**Description**

This statement controls optimizer access to abstract SQL plans.

**Plan cache eviction policy:** When an ENABLE or DISABLE ABSTRACT SQL PLAN ENTRY ALL statement is executed, any captured plans implicated by `<target>` are evicted. When a REMOVE ABSTRACT SQL PLAN ENTRY ALL statement is executed, any captured plans implicated by `<target>` with status `enabled` are evicted.

**Example**

The following example removes all abstract SQL plans:

```
ALTER SYSTEM REMOVE ABSTRACT SQL PLAN ENTRY ALL;
```

The following example enables the single abstract SQL plan with ID 9630002 for use by the optimizer:

```
ALTER SYSTEM ENABLE ABSTRACT SQL PLAN ENTRY FOR 9630002;
```

**Related Information**

SQL Plan Stability

- `ALTER SYSTEM {ADD | REMOVE} ABSTRACT SQL PLAN FILTER (System Management) [page 497]`
- `ALTER SYSTEM MIGRATE ABSTRACT SQL PLAN (System Management) [page 544]`
- `ALTER SYSTEM {START | STOP} CAPTURE ABSTRACT SQL PLAN (System Management) [page 575]`
- `ALTER SYSTEM {START | STOP} APPLY ABSTRACT SQL PLAN (System Management) [page 570]`
4.10.1.21.27 ALTER SYSTEM {ENABLE | DISABLE} ALL [ASYNCHRONOUS | SYNCHRONOUS] TABLE REPLICAS Statement (System Management)

Activates or deactivates the overall replication operation of all replication tables or of asynchronous or synchronous tables only.

Syntax - Table Replica

```
ALTER SYSTEM { ENABLE | DISABLE } ALL [ ASYNCHRONOUS | SYNCHRONOUS ] TABLE REPLICAS
```

Syntax Elements

- **ENABLE**
  Activates the overall replication operation of all specified replication tables.
- **DISABLE**
  Deactivates the overall replication operation of all specified replication tables.
- **ASYNCHRONOUS**
  Applies the operation to all asynchronous tables.
- **SYNCHRONOUS**
  Applies the operation to all synchronous tables.

Description

Activates or deactivates the overall replication operation of all replication tables or of asynchronous or synchronous tables.
Permissions

The use of this statement requires the SERVICE ADMIN privilege.

Examples

Activate all synchronous table replication.

```
ALTER SYSTEM ENABLE ALL SYNCHRONOUS TABLE REPLICAS;
```

Deactivate all asynchronous table replication.

```
ALTER SYSTEM DISABLE ALL ASYNCHRONOUS TABLE REPLICAS;
```

Deactivate all table replication.

```
ALTER SYSTEM DISABLE ALL TABLE REPLICAS;
```

4.10.1.21.28  ALTER SYSTEM {ENABLE | DISABLE} STATEMENT HINT Statement (System Management)

Enables or disables hints from a target query or statement hash.

Syntax

```
ALTER SYSTEM { ENABLE | DISABLE } STATEMENT HINT      { 
  [ ON { PROCEDURE | FUNCTION } <proc_func_name> ] FOR <target_query> 
  | FOR STATEMENT HASH <statement_hash> 
  | ALL }; 
```

Syntax Elements

```
ON { PROCEDURE | FUNCTION } proc_func_name
```

If a statement is specified as the target with the corresponding procedure/function name, the given hint is implicitly applied to the statement inside the corresponding procedure/function.

```
<proc_func_name> ::= [ <schema_name>.]<identifier>
```

```
target_query
```
Specifies the target query.

**statement_hash** Specifies the statement hash.

**ALTER SYSTEM ENABLE STATEMENT HINT FOR { <target_query> | STATEMENT HASH <statement_hash> }** Enables the statement hint specified by target_query or statement_hash only.

**ALTER SYSTEM ENABLE STATEMENT HINT ALL** Enables all hints contained in the statement hint table.

### Description

Both statements (ENABLE, DISABLE) evict all cached plans for the target SQL statements from the SQL Plan Cache.

### Example

This example enables all statement hints.

```
ALTER SYSTEM ENABLE STATEMENT HINT ALL;
```

This example disables statement hints for the SELECT query.

```
ALTER SYSTEM DISABLE STATEMENT HINT FOR SELECT * FROM AAA, BBB WHERE AAA.X = BBB.X;
```

This example enables statement hints for the statement hash 36896ef08346b8321f88449d5c5e97f8.

```
ALTER SYSTEM ENABLE STATEMENT HINT FOR STATEMENT HASH '36896ef08346b8321f88449d5c5e97f8';
```

This example disables statement hints for the statement hash 36896ef08346b8321f88449d5c5e97f8.

```
ALTER SYSTEM DISABLE STATEMENT HINT FOR STATEMENT HASH '36896ef08346b8321f88449d5c5e97f8';
```

This example disables the hint on procedure MY_PROC using the target query SELECT * FROM :TVAR AAA, BBB WHERE AAA.X = BBB.X.

```
ALTER SYSTEM DISABLE STATEMENT HINT ON PROCEDURE MY_PROC FOR SELECT * FROM :TVAR AAA, BBB WHERE AAA.X = BBB.X;
```

### Related Information

**ALTER SYSTEM {ADD | ALTER | REMOVE} STATEMENT HINT Statement (System Management) [page 493]**
4.10.1.21.29 ALTER SYSTEM {ENABLE | DISABLE} SYSTEM REPLICATION Statement (System Management)

Enables/disables an SAP HANA system to serve as an SAP HANA System Replication (HSR) primary site.

Syntax

```
ALTER SYSTEM ENABLE SYSTEM REPLICATION <site_name>
| ALTER SYSTEM DISABLE SYSTEM REPLICATION [ FORCE ]
```

Syntax Elements

- **site_name**
  Specifies a unique identifier describing the SAP HANA system that will act as the primary site.

- **FORCE**
  Disables system replication, even if there are secondary sites attached.

Description

These statements are only supported in an on-premise SAP HANA system.

The `ENABLE SYSTEM REPLICATION <site_name>` syntax configures an SAP HANA system to serve as an HSR primary site.

The `DISABLE SYSTEM REPLICATION` syntax removes the HSR configuration from the SAP HANA system. The SAP HANA system continues running, but without system replication enabled. Specify `FORCE` when you want to disable HSR while secondary sites are still attached; otherwise, the statement fails if other sites are still attached.

These statements are equivalent to configuring HSR by using the `hdbnsutil` utility. For more information about setting up a system replication primary site, see the SAP HANA Administration Guide.

Permissions

You must be in SYSTEMDB and have the SYSTEM REPLICATION ADMIN privilege to enable or disable system replication.
Examples

The following example configures an SAP HANA system to serve as an HSR primary site and assigns the site name `mySite`:

```
ALTER SYSTEM ENABLE SYSTEM REPLICATION mySite;
```

The following example removes the HSR configuration from the SAP HANA system:

```
ALTER SYSTEM DISABLE SYSTEM REPLICATION;
```

Related Information

Configure SAP HANA System Replication with hdbnsutil

4.10.1.21.30 ALTER SYSTEM ENCRYPTION CONFIGURATION Statement (System Management)

Controls encryption configuration for tenant databases.

Syntax

```
ALTER SYSTEM ENCRYPTION CONFIGURATION CONTROLLED BY { SYSTEM DATABASE | LOCAL DATABASES }
```

Syntax Elements

**SYSTEM DATABASE**

Run on the tenant database, specifies that encryption configuration is controlled by the system database. Use this syntax on the tenant database to change control over to the system database.

**LOCAL DATABASES**

Specifies that encryption configuration is controlled by the tenant databases. Use this syntax on the system database to change control over to the tenant databases.
Description

Controls encryption configuration for tenant databases.

To determine which database is controlling encryption configuration for a tenant database, check the relevant value in the CONFIGURATION_CONTROL column of the M_ENCRYPTION_OVERVIEW system view.

Examples

When run on the system database, this statement gives control of encryption configuration to the tenant databases:

```
ALTER SYSTEM ENCRYPTION CONFIGURATION CONTROLLED BY LOCAL DATABASES;
```

When run on a tenant database, the following statement gives control of encryption configuration to the system databases:

```
ALTER SYSTEM ENCRYPTION CONFIGURATION CONTROLLED BY SYSTEM DATABASE;
```

Related Information

M_ENCRYPTION_OVERVIEW System View [page 1881]

4.10.1.21.31 ALTER SYSTEM LOAD PERFTRACE Statement (System Management)

Converts a .tpt file into tables.

Syntax

```
ALTER SYSTEM LOAD PERFTRACE [FILE <file_name>]     
INTO TABLES <table_prefix> [WITH REPLACE]
```

Syntax Elements

FILE file_name
Specifies the file that raw performance data is loaded from.

```
<file_name> ::= <string_literal>
```

INTO TABLES table_prefix

Specifies the table prefix to be used with optional schema name.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The created tables are only to be used in conjunction with SAP tools or support personnel; the format of the tables is intentionally not documented for end users.</td>
</tr>
</tbody>
</table>

```
<table_prefix> ::= [ <schema_name>. ]<identifier>
<schema_name> ::= <unicode_name>
```

WITH REPLACE

Specifies that previously existing tables are removed. If the REPLACE option is not specified, then an error is thrown if tables with the specified prefix already exist.

**Description**

Tables beginning with `<table_prefix>_PERFTRACE_...` are created and filled with the content from the `.tpt` file. The `.tpt` file is loaded from the trace directory of your SAP HANA database instance. If you do not specify a file name, then `perftrace.tpt` is used.

**Example**

Load performance trace data from the `mytrace.tpt` file into tables. The created tables include `MYTRACE_PERFTRACE_INFO`, `MYTRACE_PERFTRACE_SERVICES`, and `MYTRACE_PERFTRACE_CALLS`.

```
ALTER SYSTEM LOAD PERFTRACE FILE 'mytrace.tpt' INTO TABLES mytrace WITH REPLACE;
```
4.10.1.21.32  ALTER SYSTEM LOG ENCRYPTION Statement  
(System Management)

Controls whether redo log entries are stored on disk in an encrypted or non-encrypted format and manages the encryption root keys used.

Syntax

ALTER SYSTEM LOG ENCRYPTION <encrypt_option>

Syntax Elements

encrypt_option

Specifies encryption options.

<encrypt_option> ::=  
| CREATE NEW ROOT KEY WITHOUT ACTIVATE  
| ACTIVATE NEW ROOT KEY  
| ON  
| OFF

CREATE NEW ROOT KEY WITHOUT ACTIVATE

Creates a new random encryption root key for redo log encryption with the key state PREACTIVE. You must backup the key before it is used to encrypt new redo log records.

ACTIVATE NEW ROOT KEY

Updates the state of the PREACTIVE root key to ACTIVE, and the previously ACTIVE root key to DEACTIVATED. Once activated, the new key is used for encrypting any new redo log records written to disk.

ON Turns on redo log encryption. Any new redo log records written to log volumes after redo log encryption is enabled are encrypted using the ACTIVE redo log encryption root key.

OFF Turns off redo log encryption. Any new redo log records written to log volumes after redo log encryption is turned off are written in cleartext. Existing log volumes remain encrypted.

Description

You can use this statement to enable or disable log encryption in the database, create a new random encryption root key for redo log encryption with the key state PREACTIVE, and activate an existing root key. Once activated, the new key is used to encrypt any new redo log records written to log volumes.

Execute this statement in the database for which the root key is being created.
If the system database has encryption configuration control, log encryption can only be enabled or disabled in a tenant database from the system database using ALTER DATABASE `<database_name>` LOG ENCRYPTION ON.

An error occurs if a PREACTIVE key already exists and an ALTER SYSTEM LOG ENCRYPTION CREATE NEW ROOT KEY WITHOUT ACTIVATE statement is executed. Similarly, an error occurs when a PREACTIVE key does not exist and an ALTER SYSTEM LOG ENCRYPTION ACTIVATE NEW ROOT KEY statement is executed.

You must back up the root key store between creating and activating a new root key. You can back up the root key store using BACKUP ENCRYPTION ROOT KEYS or the functions ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS and ENCRYPTION_ROOT_KEYS_EXTRACT_ALL_KEYS_FOR_DATABASE function and saving the results to a backup file.

Always back up your root keys before you enable encryption.

If there is no encryption configuration in the root key store (instance SSFS or LSS), then encryption is interpreted as OFF.

For more information about encryption in the SAP HANA database and the root key change process, see the SAP HANA Security Guide and the SAP HANA Administration Guide.

Permissions

You must have the ENCRYPTION ROOT KEY ADMIN privilege to execute this statement.

Example

Create a new log root key in the preactive state.

```
ALTER SYSTEM LOG ENCRYPTION CREATE NEW ROOT KEY WITHOUT ACTIVATE;
```

Related Information

- ALTER SYSTEM VALIDATE ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 587]
- Data and Log Volume Encryption
- Managing Data Encryption in SAP HANA
- ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 568]
- ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
- ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
4.10.1.21.33 ALTER SYSTEM LOGGING Statement (System Management)

Enables or disables logging.

Syntax

```sql
ALTER SYSTEM LOGGING <on_off>
```

Syntax Elements

`on_off`

Specifies the logging status.

```
<on_off> ::= ON | OFF
```

- **ON**
  - Enables logging.
- **OFF**
  - Disables logging.

Description

While logging is disabled, no log entries persist; only the data area is written when a savepoint is reached. This behavior can cause the loss of committed transactions when the indexserver terminates in the middle of a LOAD operation. In case of a termination, truncate and insert all data again.

**Note**

Logging mode cannot be disabled on the primary system when system replication is used.
After enabling logging, perform a savepoint to be sure that all data persists. You must also perform a data backup; otherwise, you are unable to recover this data.

ALTER SYSTEM LOGGING waits for the end (commit/rollback) of existing active write transactions. During the execution of this command, write transactions are blocked. In the event that there is a long-running write transaction, this command can fail with a lock wait timeout error. Starting a new write transaction also can be blocked and failed with the same error.

Only use this command while performing the initial load of a database. If this statement is used at other times, then data loss can occur if the indexserver is terminated while logging is disabled. Consequently, in a running system, the logging mode should be always ON and should not be modified.

Example

Disable system logging.

```
ALTER SYSTEM LOGGING OFF;
```

Enable system logging.

```
ALTER SYSTEM LOGGING ON;
```

Related Information

ALTER TABLE Statement (Data Definition) [page 589]

4.10.1.21.34 ALTER SYSTEM MIGRATE ABSTRACT SQL PLAN (System Management)

Migrates abstract SQL plans to a new SAP HANA system.

Syntax

```
ALTER SYSTEM MIGRATE ABSTRACT SQL PLAN
```

Description

Execute this statement after you complete a system upgrade.
4.10.1.21.35 ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management)

Controls whether persistent data is stored on disk in an encrypted or non-encrypted format and manages the page encryption keys and encryption root keys used.

Syntax

```
ALTER SYSTEM PERSISTENCE ENCRYPTION <encrypt_option>
```

Syntax Elements

encrypt_option

Specifies encryption options.

```
<encrypt_option> ::=   | CREATE NEW KEY
| APPLY CURRENT KEY
| CREATE NEW ROOT KEY [WITHOUT ACTIVATE]
| ACTIVATE NEW ROOT KEY
| ON
| OFF
```

CREATE NEW KEY

Creates a new random page encryption key. The new key is used to encrypt any pages written to disk during subsequent savepoint operations. The time required for this operation depends on the current workload of the SAP HANA database. No changes are made to data that has already been written to disk.
APPLY CURRENT KEY
Forces all data using old page encryption keys to be decrypted and then re-encrypted using the current page encryption key. You may wish to do this if an old key has been compromised or if you want to consolidate a set of old keys.

When new key has been created with CREATE NEW KEY and then APPLY CURRENT KEY is called, a savepoint is automatically triggered so that the new key is used to encrypt and save any in-memory pages.

CREATE NEW ROOT KEY [WITHOUT ACTIVATE]
Creates a new random encryption root key.

Specifying WITHOUT ACTIVATE creates a new version of the root key with the key state PREACTIVE. If a preactive key already exists, then an error occurs. If you do not specify WITHOUT ACTIVATE, then the specified key is created and activated. The new key is used for re-encrypting all page encryption keys after the next savepoint occurs. The point in time for the next savepoint is influenced by the current workload in the database.

ACTIVATE NEW ROOT KEY
Updates the state of the PREACTIVE root key to ACTIVE, and the previously ACTIVE root key to DEACTIVATED. Once activated, the new key is used for re-encrypting all page encryption keys after the next savepoint occurs. The point in time for the next savepoint is influenced by the current workload in the database.

ON
Specifies that encryption of data volumes should be enabled. When you switch on encryption, a random encryption key is prepared and an asynchronous background task is started that encrypts all disk data with this key.

OFF
Specifies that encryption of data volumes should be disabled. When you switch off encryption, an asynchronous background task is started which decrypts all encrypted disk data.

Description
You can use this statement to enable or disable log volume encryption, create and apply new random page encryption keys, and create and activate encryption root keys.

Execute this statement on the database where you want the encryption option to take effect.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the system database has encryption configuration control, data volume encryption can only be enabled or disabled in a tenant database from the system database using ALTER DATABASE &lt;database_name&gt; PERSISTENCE ENCRYPTION ON.</td>
</tr>
</tbody>
</table>

Only the finally written disk data is encrypted. The redo log is not affected by this command.

For more information about encryption in the SAP HANA database, see the SAP HANA Security Guide and the SAP HANA Administration Guide.

You must back up the root key store between creating and activating a new root key. You can back up the root key store using BACKUP ENCRYPTION_ROOT KEYS or the functions ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS and
ENCRYPTION_ROOT_KEYS_EXTRACT_ALL_KEYS_FOR_DATABASE function and saving the results to a backup file.

Always back up your root keys before you enable encryption.

For more information about encryption in the SAP HANA database and the root key change process, see the SAP HANA Security Guide and the SAP HANA Administration Guide.

If there is no encryption configuration in the root key store (instance SSFS or LSS), then encryption is interpreted as OFF.

Permissions

You must have the ENCRYPTION_ROOT_KEY_ADMIN or RESOURCE_ADMIN privilege to create or apply page encryption keys. You must have the ENCRYPTION_ROOT_KEY_ADMIN privilege to create and activate root keys and to enable and disable encryption.

Example

1. Create a new data volume encryption root key in the preactive state.
   
   ```sql
   ALTER SYSTEM PERSISTENCE ENCRYPTION CREATE NEW ROOT KEY WITHOUT ACTIVATE;
   ```

2. Cause a non-encrypted instance to begin the asynchronous task of encrypting disk data.

   ```sql
   ALTER SYSTEM PERSISTENCE ENCRYPTION ON;
   ```

Related Information

ALTER SYSTEM VALIDATE ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 587]
ALTER SYSTEM LOG ENCRYPTION Statement (System Management) [page 541]
ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
ALTER SYSTEM APPLICATION ENCRYPTION Statement (System Management) [page 512]
ALTER DATABASE Statement (Tenant Database Management) [page 440]
M_PERSISTENCE_ENCRYPTION KEYS System View [page 2042]
ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS Function (Security) [page 187]
ENCRYPTION_ROOT_KEYS_EXTRACT_ALL_KEYS_FOR_DATABASE Function (Security) [page 188]
Data and Log Volume Encryption
Managing Data Encryption in SAP HANA
BACKUP ENCRYPTION ROOT KEYS Statement (Backup and Recovery) [page 709]
ALTER SYSTEM SET ENCRYPTION_ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 568]
4.10.1.21.36 ALTER SYSTEM {PIN | UNPIN} SQL PLAN CACHE ENTRY Statement (System Management)

Provides a runtime mechanism to bind the target query and hints to the Hint Table to force the compilation of the target query with the hint.

Syntax

```
ALTER SYSTEM PIN SQL PLAN CACHE ENTRY <plan_id> [ WITH HINT ( <hint_items> ) ]
| ALTER SYSTEM PIN SQL PLAN CACHE ENTRY LIKE <plan_id> WITH HINT ( <hint_items> )
| ALTER SYSTEM UNPIN SQL PLAN CACHE ENTRY [ LIKE ] <plan_id>
```

Syntax Elements

- **plan_id**
  
  Specifies the identifier of the plan to be pinned.

  ```
  <plan_id> ::= <integer_literal>
  ```

  Refer to the M_SQL_PLAN_CACHE view to find the `<plan_id>` for the desired plan cache entry.

- **WITH HINT hint_items**
  
  Specifies one or more hints (comma-separated list) to apply to the query. Refer to the HINTS system view for list of available hint items.

  ```
  <hint_items> ::= <string_literal>
  ```

  Pinning a plan with a WITH HINT clause binds the target query to the HINT table.

- **PIN**
  
  PIN specifies to preserve a plan.

  ALTER SYSTEM PIN...WITH HINT clause binds the target query to the HINT table.

  ALTER SYSTEM PIN without a WITH HINT clause pins a plan to the SQL plan cache so that the pinned plan will not be evicted from the SQL plan cache, even after an ALTER SYSTEM CLEAR SQL PLAN CACHE is executed. SESSION LOCAL plans are pinned for the duration of the connection, and are evicted when the user connection is closed, regardless of its pin status.
Pinned plans are returned to being normal plans after a restart. Only hints remain after a restart.

**UNPIN**

UNPIN reverts a pinned plan to a state where it may be evicted following the normal Last Recently Used (LRU) eviction scheme, as follows:

- If the plan was compiled with a WITH HINT clause and pinned to the plan cache (ALTER SYSTEM PIN...WITH HINT), then UNPIN removes the hint, unpins the plan, and the plan will be evicted as per the LRU scheme.
- If the plan was compiled without a WITH HINT clause and pinned (ALTER SYSTEM PIN), then the plan is unpinned and will be evicted as per the LRU scheme.

**LIKE**

LIKE lets users pin and unpin queries to all the nodes in the distribution landscape (for indexservers only). If a node does not have a matching plan cache entry, then it compiles the target query with the hint and insert a new plan cache entry.

**Description**

Refer to the PINNED_SQL_PLANS view to verify the list of currently pinned queries.

If you want to unpin a plan from the SQL plan cache but keep the hint info, you can run an ALTER SYSTEM ADD STATEMENT HINT statement after unpinning.

Check whether the hints in `<hint_items>` affect the query plan by using the EXPLAIN PLAN command with the `<plan_id>` parameter.

Typical pinning scenario:

1. Identify the problematic query from the system.
2. Try to append applicable hints to the query to find the target hint.
3. Look for the corresponding cache entry from the M_SQL_PLAN_CACHE view and identify the `<plan_id>`.
4. Bind the query and the hint using the ALTER SYSTEM PIN statement. Subsequent execution of the query should use the newly compiled/cached plan.

If WITH HINT is specified, then during indexserver start-up, pinned queries are restored using information from the PINNED_SQL_PLANS view. For each pair of SQL Plan Cache key and hints in the PINNED_SQL_PLANS view, a plan with the respective hint is compiled and inserted into the SQL Plan cache. If WITH HINT is not specified, then pinned plans are no longer pinned after an indexserver restart.

**Permissions**

You must have the OPTIMIZER_ADMIN privilege to execute the statements listed in the syntax section.

You must have the OPTIMIZER_ADMIN or SYSTEM privilege to access the PINNED_SQL_PLANS view.
### Examples

The following statement pins a fictitious plan 20170203150455:

```sql
ALTER SYSTEM PIN SQL PLAN CACHE ENTRY 20170203150455;
```

The following statement unpins the fictitious plan 20170203150455:

```sql
ALTER SYSTEM UNPIN SQL PLAN CACHE ENTRY 20170203150455;
```

The following statement pins the fictitious plan 1000003, and specifies the USE_OLAP_PLAN hint:

```sql
ALTER SYSTEM PIN SQL PLAN CACHE ENTRY 1000003 WITH HINT (USE_OLAP_PLAN);
```

### Related Information

- [EXPLAIN PLAN Statement (Data Manipulation)](page 992)
- [PINNED_SQL_PLANS System View](page 1609)
- [HINTS System View](page 1585)
- [M_SQL_PLAN_CACHE System View](page 2145)

### 4.10.1.21.37 ALTER SYSTEM RECLAIM DATA SPACE Statement (System Management)

Executes row store memory reorganization at runtime.

#### Syntax

```sql
ALTER SYSTEM RECLAIM [ROW] DATA SPACE [IMMEDIATE] [<host_port>]
```

#### Syntax Elements

- **IMMEDIATE**
  
  Executes row store reorganization right away. The same for omitted case.

- **host_port**
  
  Specifies the host and port of the service to reorganize the row store.

  ```sql
  <host_port> ::= '<hostname>:<port_number>'
  ```
If this value is omitted, then the reorganization is executed on the host which the connection is made.

**Description**

Before the store is reorganized, there is an inspection of the system to analyze the current status of row store memory utilization. After that, it calculates how much memory can be reduced.

When this inspection is complete, a table list is generated. It then locks the member tables in the table list and executes actual row store memory reorganization.

The table list is determined at runtime and it can affect a number of row tables.

**Example**

Execute the row store memory reorganization.

```
ALTER SYSTEM RECLAIM DATA SPACE;
```

### 4.10.1.21.38 ALTER SYSTEM RECLAIM DATAVOLUME

**Statement (System Management)**

Frees unused space inside an SAP HANA database disk persistence.

**Syntax**

```
ALTER SYSTEM RECLAIM DATAVOLUME [SPACE] [<host_port>] <percentage_of_payload_size> <shrink_mode>
```

**Syntax Elements**

- **host_port**

  Specifies the server on which the size of the persistence should be reduced.

  `<host_port> ::= 'hostname:<port_number>'`

  If `<host_port>` is omitted, then the statement is distributed to all servers with the persistence.

  For information about hostname and port number, see notation_host_name_and_port "Notation".
**percentage_of_overload_size**

Specifies the desired percentage of the payload to which the data volume should be reduced.

```plaintext
<percentage_of_overload_size> ::= <unsigned_integer>
```

The value must not be smaller than 105. The recommended values are between 105 and 120.

**shrink_mode**

Specifies the strategy to be used to reduce the persistence size. Currently only defragment is supported.

```plaintext
<shrink_mode> ::= DEFRAGMENT
```

**Description**

This statement reduces data volume size to a percentage of payload size. This statement works in a similar way to defragmenting a hard drive. Pages that are scattered around a data volume are moved to the front of the volume, and the free space at the end of the data volume is truncated.

**Example**

The following statement defragments the persistence of all servers in the landscape and reduces them to 120% of the payload size.

```sql
ALTER SYSTEM RECLAIM DATAVOLUME 120 DEFRAGMENT;
```

**4.10.1.21.39 ALTER SYSTEM RECLAIM FLEXIBLE TABLES Statement (System Management)**

Runs garbage collection for flexible tables.

**Syntax**

```sql
ALTER SYSTEM RECLAIM FLEXIBLE TABLES
```

**Description**

Periodic garbage collection is specified when creating or altering a flexible table.
During the garbage collection process, SAP HANA scans, and then removes, dynamic columns of flexible tables that contain only NULL values. However, it may not immediately remove some of these columns, even if querying the table produces only NULL values for a particular column. This lag time is due to multiversion concurrency control (MVCC) and internal data representation. Merging the delta of flexible tables before calling the garbage collection may help bring these columns to a collectible state.

**Related Information**

CREATE TABLE Statement (Data Definition) [page 825]
ALTER TABLE Statement (Data Definition) [page 589]

**4.10.1.21.40  ALTER SYSTEM RECLAIM LOB SPACE Statement (System Management)**

Runs LOB garbage collection and removes any non-referenced LOB files.

**Syntax**

```
ALTER SYSTEM RECLAIM [ROW|COLUMN] LOB SPACE
```

**Description**

Runs the LOB garbage collection, which removes any non-referenced LOB files.

**4.10.1.21.41  ALTER SYSTEM RECLAIM LOG Statement (System Management)**

Reclaims the disk space of unused log segments when the SAP HANA database has accumulated many log segments.

**Syntax**

```
ALTER SYSTEM RECLAIM LOG
```
Description

Log segment accumulation can be caused in several ways. For example, when automatic log backup is not operational for a long period or a log savepoint is blocked for an extended time. When such issues occur, use the ALTER SYSTEM RECLAIM LOG command, but only after the root cause of the log accumulation has been fixed.

Example

Reclaim the disk space of unused log segments.

```
ALTER SYSTEM RECLAIM LOG;
```

Related Information

M_LOG_SEGMENTS System View [page 2003]

4.10.1.21.42  ALTER SYSTEM RECLAIM VERSION SPACE Statement (System Management)

Triggers the row-store garbage collector to free up memory space and enhance system responsiveness.

Syntax

```
ALTER SYSTEM RECLAIM VERSION SPACE
```

Description

You do not need to use this command if only short transactions are being executed on the system because the garbage collector is automatically triggered when a transaction ends. If there are long-running transactions on the system, then this command helps overall system performance if the m_mvcc_tables monitoring view shows a large number of row-store versions (for example, over 1M).
Example

Check the number of row-store versions on the system by using the m_mvcc_tables monitoring view:

```
SELECT * FROM m_mvcc_tables;
```

Trigger row-store garbage collection:

```
ALTER SYSTEM RECLAIM VERSION SPACE;
```

Related Information

M_MVCC.Tables System View [page 2030]

4.10.1.21.43 ALTER SYSTEM RECOMPILE SQL PLAN CACHE ENTRY Statement (System Management)

Invalidates the designated plan cache entry so that it is recompiled during the next execution time.

Syntax

```
ALTER SYSTEM RECOMPILE SQL PLAN CACHE ENTRY <plan_id>
```

Syntax Elements

`plan_id`

Specifies the entry you want to recompile.

```
<plan_id> ::= <integer_literal>
```

Obtain the `<plan_id>` from the PLAN_ID column of the M_SQL_PLAN_CACHE monitoring view.

Description

When invalidated, the entry is recompiled during the next execution time. Use this command when significant changes have occurred to the data cardinalities acted upon by the query plan.
Permissions

You must have the OPTIMIZER ADMIN privilege to execute this statement.

Example

Flag the plan cache entry with plan_id 247 for recompilation.

```
ALTER SYSTEM RECOMPILE SQL PLAN CACHE ENTRY 247;
```

Related Information

M_SQL_PLAN_CACHE System View [page 2145]

4.10.1.21.44 ALTER SYSTEM RECONFIGURE SERVICE

Statement (System Management)

Reconfigures a specified service by applying the current configuration parameters.

Syntax

```
ALTER SYSTEM RECONFIGURE SERVICE (<service_name>,<hostname>,<port_number>)
```

Syntax Element

```
service_name
```

Specifies the name of the service you wish to reconfigure. See the M_SERVICE_TYPES monitoring view topic for a list of available service types.

```
<service_name> ::= <string_literal>
```

```
hostname
```

Specifies the hostname and port number where you would like to reconfigure a service.

```
<hostname> ::= <string_literal>
<port_number> ::= <unsigned_integer>
```
You use ALTER SYSTEM RECONFIGURE SERVICE to reconfigure a specified service by applying the current configuration parameters. This command is used after changing multiple configuration parameters with the ALTER CONFIGURATION command without the RECONFIGURE option set. See the ALTER SYSTEM ALTER CONFIGURATION topic.

To reconfigure a specific service specify `<service_name>` and leave `<hostname>` and `<port_number>` empty. To reconfigure all services of a type, specify `<hostname>` and `<port_number>` and leave `<service_name>` empty. To reconfigure all services, leave all parameters empty.

Example

You use the following command to reconfigure all services on the hana.yourcompany.com host using port number 30303:

```
ALTER SYSTEM RECONFIGURE SERVICE ('','hana.yourcompany.com',30303);
```

You use the following command to reconfigure all services of type indexserver:

```
ALTER SYSTEM RECONFIGURE SERVICE ('indexserver','',0);
```

Related Information

- ALTER WORKLOAD CLASS Statement (Workload Management) [page 678]
- ALTER SYSTEM ALTER CONFIGURATION Statement (System Management) [page 500]

4.10.1.21.45 ALTER SYSTEM REFRESH RESULT CACHE Statement (System Management)

Refreshes all result cache entries related to the specified object with up-to-date results.

Syntax

```
ALTER SYSTEM REFRESH [<cache_type>] RESULT CACHE <object_name>
```
Syntax Elements

**cache_type**

Specifies the cache type. The default value is STATIC.

```cache_type ::= STATIC```

**object_name**

Specifies the object name. It can be any of the objects that has a result cache.

```<object_name>```

Those objects can be found in the system view M_RESULT_CACHE.

Description

All result cache entries related to `<object_name>` are refreshed with up-to-date results.

Examples

Refresh all static result cache entries related to `<view-object>`.

```ALTER SYSTEM REFRESH STATIC RESULT CACHE <view-object>;```

Related Information

M_RESULT_CACHE System View [page 2084]

4.10.1.21.46  ALTER SYSTEM REFRESH RESULT CACHE ENTRY Statement (System Management)

Refreshes the specified result cache entry.

Syntax

```ALTER SYSTEM REFRESH [cache_type] RESULT CACHE ENTRY cache_id;```
Syntax Elements

cache_type

Specifies the cache type. The default value is STATIC.

```
<cache_type> ::= STATIC
```

cache_id

Specifies the cache ID of the result cache entry according to the monitoring view. The M_RESULT_CACHE monitoring view provides cache IDs for static result caches.

Description

Refreshes the result cache entry specified by the `<cache_id>`. No subsequent SELECT statement is required.

Examples

Refresh the specified static result cache entry from the M_RESULT_CACHE view.

```
ALTER SYSTEM REFRESH STATIC RESULT CACHE ENTRY <cache-ID>;
```

Related Information

M_RESULT_CACHE System View [page 2084]

4.10.1.21.47 ALTER SYSTEM {REGISTER | UNREGISTER} SYSTEM REPLICATION SITE Statement (System Management)

Register or unregister a system replication secondary site.

Syntax

```
ALTER SYSTEM {REGISTER | UNREGISTER} SYSTEM REPLICATION SITE <site_name>
REMOTE HOST <remote_host_name> REPLICATION MODE <replication_mode>
[ OPERATION MODE <operation_mode> ] [ FORCE ] [ FULL REPLICA ]
```
**Syntax Elements**

**site_name** The name of the secondary site.

```plaintext
<site_name> ::= <identifier>
```

**remote_host_name** The hostname of the primary site master host.

```plaintext
<remote_host_name> ::= <string>
```

**replication_mode** The system replication mode.

```plaintext
<replication_mode> ::= 'sync'
| 'syncmem'
| 'async'
```

**operation_mode** The system replication operation mode.

```plaintext
<operation_mode> ::= 'delta_datashipping'
| 'logreplay'
| 'logreplay_readaccess'
```

[FORCE] This optional clause omits host-mapping checks.

[FULL REPLICA] This optional clause forces an initial full data shipping.

**Description**

Attach or remove a secondary site to a system replication primary system.

**Example**

The following statement registers the secondary site SiteB to the remote host `<primary_host>`.

```plaintext
ALTER SYSTEM REGISTER SYSTEM REPLICATION SITE 'SiteB'
REMOTE HOST '<primary_host>' REPLICATION MODE 'sync'
OPERATION MODE 'logreplay';
```

**Related Information**

SAP HANA System Replication
ALTER SYSTEM (ENABLE | DISABLE) SYSTEM REPLICATION Statement (System Management) [page 537]
ALTER SYSTEM (ENABLE | DISABLE) ALL [ASYNCHRONOUS | SYNCHRONOUS] TABLE REPLICA Statements (System Management) [page 534]
4.10.1.21.48 ALTER SYSTEM RELOAD FILE PSE Statement (System Management)

Load changes to the file-system PSE stores specified in the sslTrustStore, sslKeyStore, sslInternalTrustStore, and sslInternalKeyStore configuration parameters. These stores are used for external and internal secure communication via TLS, as well as for SAML trust configuration and X.509 authentication via XS application unless in-database certificate collections have been configured.

Syntax

```
ALTER SYSTEM RELOAD FILE PSE
```

Description

Changes to file-system PSE stores take effect without the need to restart the database.

Permissions

You must have the INIFILE ADMIN or SSL ADMIN system privilege.

4.10.1.21.49 ALTER SYSTEM REMOVE RESULT CACHE ENTRY Statement (System Management)

Removes the result cache entry for the specified cache ID.

Syntax

```
ALTER SYSTEM REMOVE [<cache_type>] RESULT CACHE ENTRY <cache_id>;
```

Syntax Elements

- `cache_type`
Setting `<cache_type>` to DYNAMIC ensures that the cache is refreshed every time a query on the result cache is run. The default value is STATIC.

```plaintext
<cache_type> ::= STATIC | DYNAMIC
```

**cache_id**

Specifies the cache ID of the result cache entry according to the monitoring view. The M_RESULT_CACHE monitoring view provides cache IDs for static result caches and the M_DYNAMIC_RESULT_CACHE monitoring view provides cache IDs for dynamic result caches.

**Description**

The next suitable SELECT statement rebuilds the respective result cache entry.

**Examples**

Removes the specified static result cache entry from the M_RESULT_CACHE view.

```sql
ALTER SYSTEM REMOVE STATIC RESULT CACHE ENTRY <cache-ID>;
```

Removes the specified dynamic result cache entry from the M_DYNAMIC_RESULT_CACHE view.

```sql
ALTER SYSTEM REMOVE DYNAMIC RESULT CACHE ENTRY <cache-ID>;
```

**Related Information**

ALTER SYSTEM REFRESH RESULT CACHE ENTRY Statement (System Management) [page 558]
M_RESULT_CACHE System View [page 2084]
M_DYNAMIC_RESULT_CACHE System View [page 1875]

---

4.10.1.21.50 ALTER SYSTEM REMOVE SQL PLAN CACHE ENTRY Statement (System Management)

Removes the specified plan cache entry from the SQL plan cache.

**Syntax**

```sql
ALTER SYSTEM REMOVE SQL PLAN CACHE { ENTRY <plan_id> | <where_clause> }
```
Syntax Elements

- **plan_id** Specifies a plan cache entry ID to remove.
- **where_clause** See the SELECT statement.

Description

The SQL plan cache stores plans generated by previous SQL statement executions. The SAP HANA database uses the plan cache to speed up query execution if the same SQL statement is executed multiple times. The plan cache also collects some statistics regarding plan preparation and execution.

If the specified plan ID does not exist, an error is returned.

This statement removes the specified plan cache entry ID. Plan cache entry information is available in the `M_SQL_PLAN_CACHE` monitoring view.

Permissions

You must have the OPTIMIZER ADMIN privilege.

Example

Removes the plan cache entry ID 2374637 from the SQL plan cache:

```
ALTER SYSTEM REMOVE SQL PLAN CACHE ENTRY 2374637;
```

Removes plan cache entries with an execution count of less than five:

```
ALTER SYSTEM REMOVE SQL PLAN CACHE WHERE EXECUTION_COUNT < 5;
```

Related Information

- [ALTER SYSTEM CLEAR SQL PLAN CACHE Statement (System Management)](page 522)
- [SELECT Statement (Data Manipulation)](page 1104)
- [M_SQL_PLAN_CACHE System View](page 2145)
4.10.1.21.51 ALTER SYSTEM REMOVE TRACES Statement
(System Management)

Deletes the trace files on a specified host to reduce the disk space used by large trace files.

Syntax

ALTER SYSTEM REMOVE TRACES (<hostname>, <trace_file_list>)

Syntax Elements

hostname
Specifies the name of the host where the traces are to be removed.

<trace_file_list> ::= <trace_file> [{, <trace_file>}...]
<trace_file> specifies the name of a file to be removed.

Description

Valid host name and file name combinations can be retrieved from the M_TRACEFILES system view.

When a service has a trace file open, it cannot be deleted. In this case, clear the trace file by using the ALTER SYSTEM CLEAR TRACES statement.

Example

This command deletes files on the host named HOST_NAME.

ALTER SYSTEM REMOVE TRACES ('hananode01', 'extrace.py', 'extrace.py.old');
4.10.1.21.52 ALTER SYSTEM RESET MONITORING VIEW
Statement (System Management)

Resets statistics data for the specified monitoring view.

Syntax

```
ALTER SYSTEM RESET MONITORING VIEW <view_name>
```

Syntax Elements

- **view_name**
  
  Specifies the name of the resettable monitoring view to be reset.

  ```
  <view_name> ::= [<schema_name>}.{<identifier>]
  <schema_name> ::= <unicode_name>
  ```

  Not all monitoring views can be reset using this command. The names of views that can be reset have the suffix _RESET.

Description

Use this command to define a starting point for your measurements. First, you reset the monitoring view, and then you execute an action. After the action is completed, query the _RESET version of the monitor view to get the statistical information gathered since the last reset.
Example

Reset the "SYS"."M_HEAP_MEMORY_RESET" monitoring view.

```
ALTER SYSTEM RESET MONITORING VIEW "SYS"."M_HEAP_MEMORY_RESET";
```

4.10.1.21.53 ALTER SYSTEM SAVE PERFTRACE Statement
(System Management)

Collects raw performance trace data from .prf files and saves the information into a single .tpt file.

**Note**

Performance tracing is only to be used in conjunction with SAP Support personnel. The collected performance trace data cannot be analyzed by an end user.

**Syntax**

```
ALTER SYSTEM SAVE PERFTRACE [INTO FILE <file_name>]
```

**Syntax Elements**

`INTO FILE file_name`

Specifies the file where raw performance data is saved.

`<file_name> ::= <string_literal>`

**Description**

The .tpt file is saved in the trace directory of your SAP HANA database instance. If you do not specify a file name, then perftrace.tpt is used.

Use the M_JOB_PROGRESS system view to monitor the saving of a trace file. The save job is shown as Save PerfTrace in the system view.

Saving a performance trace can take some time. You can cancel the job shown in M_JOB_PROGRESS system view by using the ALTER SYSTEM CANCEL [WORK IN] SESSION statement.
Example

Save raw performance trace data into the `mytrace.tpt` file.

```
ALTER SYSTEM SAVE PERFTRACE INTO FILE 'mytrace.tpt';
```

Related Information

- ALTER SYSTEM (START | STOP) PERFTRACE Statement (System Management) [page 578]
- M_PERFTRACE System View [page 2041]
- M_JOB_PROGRESS System View [page 1948]
- Performance Trace View Diagnostic Files in the SAP HANA Database Explorer
- ALTER SYSTEM LOAD PERFTRACE Statement (System Management) [page 539]
- M_PERFTRACE System View [page 2041]
- M_JOB_PROGRESS System View [page 1948]
- ALTER SYSTEM CANCEL [WORK IN] SESSION Statement (System Management) [page 516]

**4.10.1.21.54  ALTER SYSTEM SAVEPOINT Statement (System Management)**

Executes a database checkpoint on the persistence manager.

**Syntax**

```
ALTER SYSTEM SAVEPOINT
```

**Description**

A savepoint is a point in time when a complete consistent image of the database is persisted to the disk. The consistent image can be used to restart the database.

Normally a savepoint is executed periodically as configured by the configuration parameter `savepoint_interval_s` in the [persistence] section. For special (normal test) purposes, the savepoint can be disabled. In this case, use this command to manually execute a savepoint.

The views associated with the ALTER SYSTEM SAVEPOINT statement are: M_SAVEPOINTS, M_SAVEPOINT_STATISTICS, M_SAVEPOINT_STATISTICS_RESET, and HOST_SAVEPOINTS.
This statement should not be confused with the SAVEPOINT statement, which performs a transactional savepoint.

**Example**

Execute a savepoint on the persistence manager.

```
ALTER SYSTEM SAVEPOINT;
```

**Related Information**

- `M_SAVEPOINT_STATISTICS System View [page 2097]`
- `M_SAVEPOINT_STATISTICS_RESET System View [page 2101]`
- `M_SAVEPOINTS System View [page 2093]`
- `HOST_SAVEPOINTS View (Embedded Statistics Service)`
- `SAVEPOINT Statement (Transaction Management) [page 1103]`

### 4.10.1.21.55 ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management)

Sets a password for encrypting root key backups.

**Syntax**

```
ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD <password>
```

**Syntax Elements**

- **password**
  
  The specified password overwrites any existing password.
**Description**

This statement must be executed in the database for which the root key backup password is being specified. The password is stored in the encryption root key store (instance SSFS or local secure store (LSS)). For more information about encryption in the SAP HANA database, see the *SAP HANA Security Guide* and the *SAP HANA Administration Guide*.

**Permissions**

You must have the ENCRYPTION ROOT KEY ADMIN privilege to execute this statement.

**Example**

Set a password that is used to protect root key backups.

```
ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD My1Chosen2Password3;
```

For information on how to backup and restore the encryption root key store (instance SSFS or local secure store (LSS)) by using the hdbnsutil utility, see the *SAP HANA Administration Guide*.

**Related Information**

- ALTER SYSTEM VALIDATE ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 587]
- M_PERSISTENCE_ENCRYPTION_KEYS System View [page 2042]
- Server-Side Data Encryption Services
- Managing Data Encryption in SAP HANA
- ALTER SYSTEM LOG ENCRYPTION Statement (System Management) [page 541]
- ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
- ALTER SYSTEM APPLICATION ENCRYPTION Statement (System Management) [page 512]
- ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
- ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS Function (Security) [page 187]
- ENCRYPTION_ROOT_KEYS System View [page 1558]
- APPLICATION_ENCRYPTION_KEYS System View [page 1498]
- M_ENCRYPTION_OVERVIEW System View [page 1881]
- BACKUP ENCRYPTION ROOT KEYS Statement (Backup and Recovery) [page 709]
- RECOVER ENCRYPTION ROOT KEYS Statement (Backup and Recovery) [page 1079]
4.10.1.21.56 ALTER SYSTEM {START | STOP} APPLY ABSTRACT SQL PLAN (System Management)

Starts or stops matching executed queries with captured abstract SQL plans.

Syntax

```
ALTER SYSTEM { START | STOP } APPLY ABSTRACT SQL PLAN
```

Syntax Elements

**START**

Loads all valid abstract SQL plans from the ABSTRACT_SQL_PLANS system table into the plan stability manager.

Wherever possible, any newly compiled SELECT statements are based on the abstract SQL plans. Queries are checked against the stored abstract plans and the abstract plan is used to generate an execution plan. When the plan is invalidated during generating an execution plan due to an error, the reason for the invalidation will be stored in the NOTES column in ABSTRACT_SQL_PLANS.

**STOP**

Stops the query optimizer from checking queries against stored abstract plans. STOP also unloads all valid abstract SQL plans in the ABSTRACT_SQL_PLANS system table from the plan cache; the optimizer stops using abstract SQL plans to execute queries.

Description

Starts or stops matching executed queries with captured abstract SQL plans.

**Plan cache eviction policy:** When this statement is executed, any stored plans with the status enabled are evicted.

Related Information

SQL Plan Stability

ALTER SYSTEM {ADD | REMOVE} ABSTRACT SQL PLAN FILTER (System Management) [page 497]
ALTER SYSTEM {ENABLE | DISABLE | REMOVE} ABSTRACT SQL PLAN (System Management) [page 532]
ALTER SYSTEM MIGRATE ABSTRACT SQL PLAN (System Management) [page 544]
ALTER SYSTEM {START | STOP} CAPTURE ABSTRACT SQL PLAN (System Management) [page 575]
4.10.1.21.57 ALTER SYSTEM {START | STOP | SAVE | CLEAR} KERNEL PROFILER Statement (System Management)

Manages the operation of the Kernel Profiler.

Syntax

Alter System { <start_profiler> | <stop_profiler> | <save_profiler> | <clear_profiler> }

Syntax Element

start_profiler

Starts the Kernel Profiler.

<start_profiler> ::= START KERNEL PROFILER
[ <location> ]
[ <profile_by> ]
[ <memory_limit> ]
[ <sampling_interval> ]

location

Restricts profiling to the specified location. See the definition of <location> later in this list.

profile_by

Sets the scope of the profiling.

<profile_by> ::= ( <user_name>
| <application_user_name>
| <connection_id>
| <root_statement_hash>
| <traceprofile_name>
)

user_name
Restricts profiling to the specified SQL user name.

```sql
<user_name> ::= USER <simple_identifier>
```

**application_user_name**

Restricts profiling to the specified application user name.

```sql
,application_user_name
<application_user_name> ::= APPLICATIONUSER <string_literal>
```

**connection_id**

Restricts profiling to the specified connection ID.

```sql
,connection_id
<connection_id> ::= SESSION <numeric_literal>
```

**root_statement_hash**

Restricts profiling to the specified root statement hash.

```sql
,root_statement_hash
root_statement_hash ::= ROOT_STATEMENT_HASH '<string_literal>'
```

**traceprofile_name**

Restricts profiling to the specified user-specific trace profile.

```sql
,traceprofile_name
traceprofile_name ::= TRACEPROFILE <string_literal>
```

**memory_limit**

Sets an approximate memory limit (in bytes) for the Kernel Profiler. If not specified, the default is 0 (no limit).

```sql
,memory_limit
<memory_limit> ::= MEMORY LIMIT <numeric_literal>
```

**sampling_interval**

Periodically stores profiler samples, according to specified interval (in milliseconds). If not specified, the default is 1 millisecond.

```sql
,sampling_interval
<sampling_interval> ::= SAMPLING INTERVAL <numeric_literal>
```

**stop_profiler**

Stops the Kernel Profiler (but does not free up the allocated memory).

```sql
,stop_profiler
<stop_profiler> ::= STOP KERNEL PROFILER [ <location> ]
```

**save_profiler**

Stops the Kernel Profiler, if it is running, saves the data to the trace directory, and clears the allocated memory.

```sql
,save_profiler
<save_profiler> ::= SAVE KERNEL PROFILER [ <location> ] [ <filter_list> ] [ <into_files> ]
<filter_list> ::= FOR CALLSTACKS <string_literal> [, <string_literal> [
[,] ... ]]
<into_files> ::= INTO { [ CPU FILE <file_name_cpu> ] [ WAIT FILE 
<file_name_wait> ] )
<file_name_cpu> ::= <string_literal>
```
location
Restricts profiling to the specified location. See the definition of <location> later in this topic.

filter_list
Filters the profiling output to the specified call stacks.

into_files
Writes Kernel Profiler output to the specified file(s) in the trace directory of your SAP HANA database instance.

If the INTO clause is not specified, then the default names of files it creates are kernel_profiler_cpu.dot and kernel_profiler_wait.dot. However, if only one file type is specified, only this file is created (the other file is not created with its default file name).

If you specify a file name without the prefix kernel_profiler_ or suffix _cpu.dot/_wait.dot, these are added to the file name.

You cannot specify a path for the files. Also, files of the same name are overwritten.

clear_profiler
Stops the Kernel Profiler, and clears the allocated memory.

<clear_profiler> ::= CLEAR KERNEL PROFILER [ <location> ]

<location> restricts profiling to the specified location. See the definition of <location> later in this topic.

location
Restricts the Kernel Profiler to the specified location.

<location> ::= AT [ LOCATION ] <host_and_port>
<host_and_port> ::= ( ( <host> , <port> ) | '<host>':<port> )
<number> ::= <numeric_literal>

Description

The Kernel Profiler is a tool which collects information about CPU consumption and wait times for SAP HANA’s various internal processes. This information can be helpful in support scenarios.

Only one instance of the Kernel Profiler can be active at a time using the ALTER SYSTEM statement, which can be monitored with the M_KERNEL_PROFILER system view.

Permissions

You must have the RESOURCE ADMIN or TRACE ADMIN system privilege to execute this statement.
Examples

The following statement clears all previously collected Kernel Profiler data.

```
ALTER SYSTEM CLEAR KERNEL PROFILER;
```

The following statement starts profiling for the SYSTEM user, sets a memory limit of 1 GB, and sets a sampling interval of 2 milliseconds.

```
ALTER SYSTEM START KERNEL PROFILER USER SYSTEM MEMORY LIMIT 1073741824 SAMPLING INTERVAL 2;
```

The following statement stops profiling, saves the data for the specified call stacks into the trace directory, and then frees the memory that was used for the profiling data.

```
ALTER SYSTEM SAVE KERNEL PROFILER FOR CALLSTACK 'Execution::JobExecWatchdog::run','System::ProcessorInfo::getCurrentProcessorIndex' INTO CPU FILE 'cpu.dot' WAIT FILE 'wait.dot';
```

The following statement starts profiling at host:port `ab1234:30003` and filters for the user-specific trace (database trace) profile `MYTRACEPROFILE`.

```
ALTER SYSTEM START KERNEL PROFILER AT 'ab1234:30003' TRACEPROFILE 'MYTRACEPROFILE';
```

The following statement stops profiling and clears the profiling data for the specified location `ab1234:30003`. In this case, because the Kernel Profiler is being stopped, and the data cleared, no data will be saved.

```
ALTER SYSTEM CLEAR KERNEL PROFILER AT LOCATION 'ab1234:30003';
```

The following example shows how to profile using the root statement hash:

```
ALTER SYSTEM START KERNEL PROFILER ROOT_STATEMENT_HASH 'a68e7882f87a2ec501604ce6feb7dee?';
SELECT TOP 10 * from M_SERVICE_THREADS ORDER BY DURATION;                    --has this hash (maybe execute several times)
ALTER SYSTEM STOP KERNEL PROFILER;
SELECT * FROM SYS.M_KERNEL_PROFILER;                                           --check for root_statement_hash column and also if SAMPLE_COUNT > 0
ALTER SYSTEM SAVE KERNEL PROFILER;                                            --if SAMPLE_COUNT > 0
ALTER SYSTEM CLEAR KERNEL PROFILER;
```

Related Information

- `M KERNEL PROFILER System View [page 1956]`
- Kernel Profiler
- Diagnostic Files and Logs
- Database Trace (Basic, User-Specific, and End-to-End)
4.10.1.21.58 ALTER SYSTEM {START | STOP} CAPTURE ABSTRACT SQL PLAN (System Management)

Starts or stops capturing abstract SQL plans for queries that are executed against the database.

Syntax

```
ALTER SYSTEM START CAPTURE ABSTRACT SQL PLAN
    [ FOR USER <user_name_list> ]
    [ WITH SQL PLAN CACHE ]
| ALTER SYSTEM STOP CAPTURE ABSTRACT SQL PLAN
```

Syntax Element

START

Starts the collection of abstract SQL plans for newly compiled queries.

FOR USER user_name_list

Captures and stores abstract SQL plans for newly compiled queries that are executed by the specified users.

```
<user_name_list>::= <user_name>[, <user_name>]...
```

WITH SQL PLAN CACHE

Controls whether cached SQL queries in the SQL plan cache are captured. Specify this clause to capture newly compiled SQL queries and capture cached SQL queries. Otherwise, only newly compiled SQL queries are captured (the default).

This clause recompiles the queries that are currently cached in the system.

STOP

Stops the collection of abstract SQL plans for newly compiled queries.

If a background job is still capturing plans when this statement is executed, then plan stability does not capture any more plans. The stop capture takes effect after the current statement that is being captured is finished.

Description

The capture process runs in the background while selected queries are executed.
Example

Start capturing abstract SQL plans for all database users:

```sql
ALTER SYSTEM START CAPTURE ABSTRACT SQL PLAN;
```

Capture abstract SQL plans only for the specified users:

```sql
ALTER SYSTEM START CAPTURE ABSTRACT SQL PLAN FOR USER TESTUSER1, TESTUSER2;
```

Related Information

SQL Plan Stability

- `ALTER SYSTEM {ENABLE | DISABLE | REMOVE} ABSTRACT SQL PLAN` (System Management) [page 532]
- `ALTER SYSTEM MIGRATE ABSTRACT SQL PLAN` (System Management) [page 544]
- `ALTER SYSTEM {START | STOP} APPLY ABSTRACT SQL PLAN` (System Management) [page 570]
- `ALTER SYSTEM UPDATE ABSTRACT SQL PLAN` Statement (System Management) [page 586]
- `ABSTRACT_SQL_PLANS` System View [page 1477]
- `M_ABSTRACT_SQL_PLAN_OVERVIEW` System View [page 1730]
- `M_ABSTRACT_SQL_PLAN_STATISTICS` System View [page 1731]

4.10.1.21.59 ALTER SYSTEM {START | STOP | CLEAR} SQLSCRIPT PLAN PROFILER Statement (System Management)

Starts, stops, and clears the SQLScript Plan Profiling feature.

Syntax

```sql
ALTER SYSTEM { START | STOP | CLEAR } SQLSCRIPT PLAN PROFILER [ <filter> ]
<filter> ::= FOR SESSION <session_id> | FOR PROCEDURE <procedure_name>
```

Syntax Element

- **START**
  
  Enables the SQLScript plan profiler.

- **STOP**
  
  Disables the SQLScript plan profiler.
**CLEAR** Clears the results of the SQLScript plan profiler from the M_SQLSCRIPT_PLAN_PROFILER_RESULTS system view.

**filter**
Specifies the scope of plan profiling - either a session, or a procedure. When a filter is applied, only the statements executed in the specified scope is profiled. The default is the entire system.

**Description**

Use this statement to enable SQLScript plan profiling for a session or procedure, or to stop SQLScript plan profiling, or clear the results from profiling.

You can query SQLScript plan profiling results from the M_SQLSCRIPT_PLAN_PROFILER_RESULTS system view. To display a list of the sessions and procedures that are being profiled using the SQLScript plan profiler, query the M_SQLSCRIPT_PLAN_PROFILERS system view.

You can also enable the SQLScript plan profiler for a procedure by adding the SQLSCRIPT_PLAN_PROFILER hint to a CALL statement (for example, `CALL P1 WITH HINT (SQLSCRIPT_PLAN_PROFILER)`).

**Permissions**

You must have the TRACE ADMIN or OPTIMIZER ADMIN system privilege to execute this statement.

**Example**

The following example starts the SQLScript plan profiler:

```
ALTER SYSTEM START SQLSCRIPT PLAN PROFILER;
```

**Related Information**

SAP HANA SQLScript Reference
M_SQLSCRIPT_PLAN_PROFILER_RESULTS System View [page 2139]
M_SQLSCRIPT_PLAN_PROFILERS System View [page 2137]
4.10.1.21.60  ALTER SYSTEM {START | STOP} PERFTRACE
Statement (System Management)

Starts or stops performance tracing.

**Note**
Performance tracing is only to be used in conjunction with SAP Support personnel. The collected performance trace data cannot be analyzed by an end user.

**Syntax**

```
ALTER SYSTEM START PERFTRACE [ <user_name> ] [ <application_user_name> ]
[ <application_name> ] [ <passport_level> ]
[ PLAN_EXECUTION ] [ FUNCTION_PROFILER ] [ DURATION <duration_seconds> ]
[ <root_statement_hash> ]
| ALTER SYSTEM STOP PERFTRACE
```

**Syntax Element**

- **user_name**
  Restricts performance trace collection to an SQL user name.
  
  `<user_name> ::= USER <simple_identifier>`

- **application_user_name**
  Restricts performance trace collection to the application user name.
  
  `<application_user_name> ::= APPLICATIONUSER <string_literal>`

- **application_name**
  Restricts performance trace collection to the application name.
  
  `<application_name> ::= APPLICATION <string_literal>`

- **passport_level**
  Specifies the level of trace data to be collected.
  
  `<passport_level> ::= PASSPORT_TRACELEVEL { MEDIUM | HIGH }`

  `<passport_level>` filters the amount of data collected in end-to-end scenarios.

- **PLAN_EXECUTION**
  Specifies that plan execution details should be traced.

- **FUNCTION_PROFILER**
Specifies that function-level details should be traced.

**duration_seconds**

Specifies the number of seconds to run the performance trace for. If not specified, the trace runs until the ALTER SYSTEM STOP PERFTRACE statement is executed.

```<duration_seconds> ::= <numeric_literal>```

After this period expires, the performance trace is automatically stopped. If you do not specify this parameter, then you must stop the performance trace with the ALTER SYSTEM STOP PERFTRACE statement.

**root_statement_hash**

Restricts performance trace collection to the statement identified by the root statement hash.

```<root_statement_hash> ::= ROOT_STATEMENT_HASH <string_literal>```

For statements in stored procedures, the root statement hash is the statement hash of the stored procedure call. The perftrace filter compares the given root statement hash with the root statement hash of the statements. This will also trace statements running within a stored procedure.

**Description**

Only one performance trace can be active at a time.

After stopping the trace, collect and save the performance trace data with the ALTER SYSTEM SAVE PERFTRACE statement.

**Example**

Start performance tracing for the user sql_user on application user name app_user for application app_name, and specify that plan execution and function-level detail should be traced.

```
ALTER SYSTEM START PERFTRACE USER sql_user APPLICATIONUSER app_user APPLICATION app_name PLAN_EXECUTION FUNCTION_PROFILER;
```

Start performance tracing that is filtered by the statement hash 21db1b17968de7861e6c97645f72963b.

```
ALTER SYSTEM START PERFTRACE ROOT_STATEMENT_HASH '21db1b17968de7861e6c97645f72963b';
```

Stop an active performance trace.

```
ALTER SYSTEM STOP PERFTRACE;
```
4.10.1.21.61 ALTER SYSTEM {START | STOP} SQLSCRIPT CODE COVERAGE Statement (System Management)

Starts and stops a SQLScript code coverage session for functions and procedures.

Syntax

To start SQLScript code coverage:

```
ALTER SYSTEM START SQLSCRIPT CODE COVERAGE
  [ FOR DEBUG TOKEN <token_id> ]
  [ FOR USER <user_id> ]
  [ FOR APPLICATION USER <application_user_id> ]
  [ FOR SESSION <session_id> ]
```

To stop SQLScript code coverage:

```
ALTER SYSTEM STOP SQLSCRIPT CODE COVERAGE
```

Syntax Elements

- **token_id**
  - Specifies the token that the code coverage applies to.
- **user_id**
  - Specifies the database user ID that the code coverage applies to.
- **application_user_id**
  - Specifies the ID of the application user that the code coverage applies to.
**session_id**

Specifies the ID of the session that the code coverage applies to.

**Description**

See the SAP HANA SQLScript Reference Guide for more information on code coverage support for functions and procedures in SQLScript.

SAP HANA stores the results of a code coverage session in the M_SQLSCRIPT_CODE_COVERAGE_RESULTS monitoring view and stores the definitions of objects that were used during a code coverage session in the M_SQLSCRIPT_CODE_COVERAGE_OBJECT_DEFINITIONS monitoring view.

Select from the monitoring views at any time, and from any column, you are interested in after starting code coverage. However, the full content of code coverage run is visible only after the query triggered in the second session (which is being covered) finishes (described in the second example, below).

The content in the monitoring views is overwritten in these views each time you stop a SQLScript code coverage session and start a new one. Since the data is temporary, copy or export the content from these views to retain data recorded by a SQLScript code coverage session before executing ALTER SYSTEM STOP SQLSCRIPT CODE COVERAGE.

You must have at least two connections for code coverage. In the first session you execute the codes on which you run code coverage, and in the second session you start the code coverage for a specific connection ID to record the coverage.

**Permissions**

You must have the EXECUTE, DEBUG, and ATTACH_DEBUGGER privileges to perform code coverage.

**Example**

SAP HANA requires two sessions to perform the code coverage. The examples below use session A to execute the code on which you run code coverage, and session B starts the code coverage for a specific connection ID to record the coverage.

1. In either session, create the limitedLoop and dummy_proc procedures:

```sql
CREATE PROCEDURE limitedLoop() AS
BEGIN
  DECLARE i BIGINT := 0;
  LOOP
    i := i + 1;
    IF :i > 27 THEN
      BREAK;
    END IF;
  END LOOP;
END;
CREATE PROCEDURE dummy_proc() AS
```
BEGIN
    SELECT * FROM DUMMY;
    CALL limitedLoop();
END;

2. From session A, issue this to determine the connection ID:

   SELECT SESSION_CONTEXT('CONN_ID') FROM DUMMY;

3. In session B, start code coverage by using the connection ID of the user who is executing the code in session A (this example uses a connection ID of 203247):

   ALTER SYSTEM START SQLSCRIPT CODE COVERAGE FOR SESSION '203247';

4. From session A, call the dummy_proc procedure:

   CALL dummy_proc();

5. From session B, view the code coverage by querying the M_SQLSCRIPT_CODE_COVERAGE_RESULTS and M_SQLSCRIPT_CODE_COVERAGE_OBJECT_DEFINITIONS monitoring views

   SELECT * FROM M_SQLSCRIPT_CODE_COVERAGE_RESULTS;
   SELECT * FROM M_SQLSCRIPT_CODE_COVERAGE_OBJECT_DEFINITIONS;

   If required, store the contents of the monitoring views for later reference (this can be a regular or a local temporary table):

   CREATE LOCAL TEMPORARY TABLE "#SomeTableName" AS (SELECT * FROM M_SQLSCRIPT_CODE_COVERAGE_RESULTS) WITH DATA;

6. From session B, disable the code coverage (this also clears the existing code coverage):

   ALTER SYSTEM STOP SQLSCRIPT CODE COVERAGE;

Related Information

SAP HANA SQLScript Reference
CREATE PROCEDURE Statement (Procedural) [page 776]
CREATE FUNCTION Statement (Procedural) [page 752]
M_SQLSCRIPT_CODE_COVERAGE_RESULTS System View [page 2136]
M_SQLSCRIPT_CODE_COVERAGE_OBJECT_DEFINITIONS System View [page 2135]
4.10.1.21.62 ALTER SYSTEM START DATABASE Statement
(Tenant Database Management)

Starts the specified database.

Syntax

ALTER SYSTEM START DATABASE <database_name> [ FROM FALLBACK SNAPSHOT ]

Syntax Elements

database_name

Specifies the database to start.

<database_name> ::= <identifier>

FROM FALLBACK SNAPSHOT

Starts the tenant database using the fallback snapshot. This allows you to restore the tenant database to the state it was in when the fallback snapshot was created. For more information about fallback snapshots, see the SAP HANA Tenant Databases Guide.

Permissions

You must have the DATABASE ADMIN or DATABASE START privilege to use this statement.

Example

Start the my_database database.

ALTER SYSTEM START DATABASE my_database;

Related Information

ALTER DATABASE Statement (Tenant Database Management) [page 440]
Create a Fallback Snapshot
4.10.1.21.63  ALTER SYSTEM STOP DATABASE Statement
(Tenant Database Management)

Stops the specified database.

Syntax

```sql
ALTER SYSTEM STOP DATABASE <database_name> [ IMMEDIATE [ WITH COREFILE ] ]
```

Syntax Elements

database_name

Specifies the name of database to stop.

```sql
<database_name> ::= <identifier>
```

IMMEDIATE

Immediately stops the database without waiting for a regular shutdown.

WITH COREFILE

Writes a core dump file.

Description

Executing this statement stops all of the services that belong to the database on all hosts. The database is stopped immediately even if users are connected. Open transactions are aborted and rolled back; no save point operation is forced. It is not possible to back up a stopped database. This statement can only be issued from the SYSTEM database (systemDB) and is only available for SAP HANA tenant databases.

Permissions

You must have the DATABASE ADMIN or DATABASE STOP privilege to use this statement.
Example

Stop the database named my_database.

```
ALTER SYSTEM STOP DATABASE my_database;
```

4.10.1.21.64  ALTER SYSTEM STOP SERVICE Statement
(System Management)

Stops single or multiple services on the designated host.

Syntax

```
ALTER SYSTEM STOP SERVICE <host_port> [IMMEDIATE [WITH COREFILE]]
```

Syntax Element

```
host_port
```

Specifies the host and port of the service to be stopped.

```
<host_port> ::=    <hostname>:<port_number> | ('<host_name>',<port_number>)
```

IMMEDIATE

Immediately stops the service without waiting for regular shutdown.

WITH COREFILE

Writes a core dump file.

Description

Typically a service you stop is automatically restarted by the SAP HANA database system.

You should use this command after changing a configuration parameter, which requires a restart of the service.
Example

Stop a service running on host hdb1.yourcompany.com on port 30303.

```
ALTER SYSTEM STOP SERVICE 'hdb1.yourcompany.com:30303';
```

4.10.1.21.65 ALTER SYSTEM UPDATE ABSTRACT SQL PLAN
Statement (System Management)

Updates the location information for abstract SQL plans.

Syntax

```
ALTER SYSTEM UPDATE ABSTRACT SQL PLAN
   SET LOCATION <location_info_map_list> ALL
```

Syntax Element

```
SET LOCATION location_info_map_list
```

Specifies the location of the abstract SQL plan. When you specify the location, then the location information of the target abstract SQL plan is updated with the specified location information map and then imported into the system.

```
<location_info_map_list> ::=      <from_location> TO <to_location>[, <location_info_map_list> ... ]
<from_location> ::= '<host_name>:<port_number>'
<to_location> ::= '<host_name>:<port_number>'
```

ALL Updates the location information for all abstract SQL plans.

Description

Updates the location information for abstract SQL plans.
Examples

The following statement updates the location of all abstract SQL plans:

```
ALTER SYSTEM UPDATE ABSTRACT SQL PLAN
    SET LOCATION 'selibm57:30103' TO 'selibm58:30103' ALL;
```

Related Information

SQL Plan Stability
ALTER SYSTEM {ADD | REMOVE} ABSTRACT SQL PLAN FILTER (System Management) [page 497]
ALTER SYSTEM {ENABLE | DISABLE | REMOVE} ABSTRACT SQL PLAN (System Management) [page 532]
ALTER SYSTEM MIGRATE ABSTRACT SQL PLAN (System Management) [page 544]
ALTER SYSTEM (START | STOP) CAPTURE ABSTRACT SQL PLAN (System Management) [page 575]
ALTER SYSTEM (START | STOP) APPLY ABSTRACT SQL PLAN (System Management) [page 570]
ABSTRACT_SQL_PLANS System View [page 1477]
M_ABSTRACT_SQL_PLAN_OVERVIEW System View [page 1730]
M_ABSTRACT_SQL_PLAN_STATISTICS System View [page 1731]

4.10.1.21.66 ALTER SYSTEM VALIDATE ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management)

Verifies that the encryption root key backup password that is specified on the command line is the same as the one stored in the encryption root key store.

Syntax

```
ALTER SYSTEM VALIDATE ENCRYPTION ROOT KEYS BACKUP PASSWORD <password>
```

Syntax Elements

password

The specified password is compared with the password stored in the encryption root key store.
Description

This statement must be executed in the database for which the root key backup password is being specified. The password is stored in the encryption root key store (instance SSFS or local secure store (LSS)). For more information about encryption in the SAP HANA database, see the SAP HANA Security Guide and the SAP HANA Administration Guide.

Permissions

You must have the ENCRYPTION ROOT KEY ADMIN privilege to execute this statement.

Example

Check that the password My1Chosen2Password3 is the same as the encryption root key backup password that is stored in the encryption root key store.

```
ALTER SYSTEM VALIDATE ENCRYPTION ROOT KEYS BACKUP PASSWORD 'My1Chosen2Password3';
```

Related Information

M_PERSISTENCE_ENCRYPTION_KEYS System View [page 2042]
ALTER SYSTEM VALIDATE ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 587]
Managing Data Encryption in SAP HANA
ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 568]
ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
ALTER SYSTEM APPLICATION ENCRYPTION Statement (System Management) [page 512]
ALTER SYSTEM LOG ENCRYPTION Statement (System Management) [page 541]
ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
ENCRYPTION_ROOT_KEY_EXTRACT_KEYS Function (Security) [page 187]
ENCRYPTION_ROOT_KEYS System View [page 1558]
APPLICATION_ENCRYPTION_KEYS System View [page 1498]
M_ENCRYPTION_OVERVIEW System View [page 1881]
4.10.1.22 ALTER TABLE Statement (Data Definition)

Alters a base or temporary table. See the ALTER VIRTUAL TABLE statement for altering virtual tables.

Syntax

```
ALTER TABLE <table_name>
  <column_clauses>
  <system_versioning_configuration>
  <add_drop_application_time_period>
  <lob_reorganize_clause>
  <clear_column_join_data_statistics_clause>
  <constraint_clause>
  <primary_key_clauses>
  <preload_clause>
  <table_conversion_clause>
  <association_clauses>
  <with_annotation_clause>
  <partition_clauses>
  <table_load_unit_clause>
  <alter_persistent_memory_spec_clause>
  <replica_clauses>
  < redo_log_option>
  <auto_merge_option>
  <unload_priority>
  <schema_flexibility_option>
  < trigger_option>
  <group_option_clauses>
  <row_order_clauses>
  < unused_retention_period>
  < reclaim_data_space_clause>
  <series_reorganize_clause>
  < owner_to_clause>
  <alter_mask_settings_clause>
  <clientside_encryption_clauses>
  <set_movable_clause>
  <convert_index_type>
  < numa_node_preference_clause>
```

Syntax Elements

table_name

Specifies the identifier of the table to be altered, with optional schema name.

```
<table_name> ::= [ <schema_name>.] <identifier>
<schema_name> ::= <identifier>
```

column_clauses

Defines the column operations you can perform.

```
<column_clauses> ::= <add_columns_clause>
  <alter_columns_clause>
  <drop_columns_clause>
```
### add_columns_clause

Add one or more columns to the specified table.

```sql
<add_columns_clause> ::= ADD ( <column_list> )
<column_list> ::= <column_specification> [, <column_specification> [,...] [ ONLINE { PREFERRED }] ]
<column_specification> ::= <column_name> { [ <column_definition> ] [ <column_constraint_short> ] [ COMMENT <string_literal> ] [ ONLINE ] [ DETERMINISTIC ] [ CLIENTSIDE ENCRYPTION ON WITH <column_encryption_key_name> [ RANDOM | DETERMINISTIC ] [ <add_column_load_unit> ]
<add_column_load_unit> ::= <column_name> <add_load_unit>
```

### column_definition

Specifies a definition for a column.

```sql
<column_definition> ::= [ { <data_type> | <lob_data_type> } <ddic_data_type> ]
[ <default_value_clause> ]
[ <col_gen_as_expression> ]
[ <col_gen_as_ident> ]
[ <col_calculated_field> ]
[ <schema_flexibility> ]
[ <fuzzy_search_index> ]
[ <fuzzy_search_mode> ]
[ <alter_persistent_memory_spec_clause> ]
```

### data_type

Specifies the column data types.

```sql
<data_type> ::= DATE
| TIME
| SECONDDATE
| TIMESTAMP
| TINYINT
| SMALLINT
| INTEGER
| BIGINT
| SMALLDECIMAL
| REAL
| DOUBLE
| TEXT
| BINTEXT
| VARCHAR [ (<unsigned_integer>) ]
| NVARCHAR [ (<unsigned_integer>) ]
| ALPHANUM [ (<unsigned_integer>) ]
| SHORTTEXT [ (<unsigned_integer>) ]
| VARBINARY [ (<unsigned_integer>) ]
| DECIMAL [ (<unsigned_integer> [ , <unsigned_integer> ] ) ]
| FLOAT [ (<unsigned_integer>) ]
| BOOLEAN
```

For tables with time-selection partitioning, the data type for the time-selection column must be NVARCHAR(8).

### lob_data_type

Specifies a LOB data type.

```sql
<lob_data_type> ::=<lob_type_name> [MEMORY THRESHOLD <memory_threshold_value>]
<lob_type_name> ::= PUBLIC
```

---

SAP HANA SQL Reference Guide for SAP HANA Platform

590 PUBLIC
<memory_threshold_value> controls whether LOB data should be stored in memory, according to the following conditions:

- If <memory_threshold_value> is not provided, then a hybrid LOB with a memory threshold of 1000 bytes is created by default.
- If <memory_threshold_value> is provided and its LOB size is bigger than the memory threshold value, then LOB data is stored on disk.
- If <memory_threshold_value> is provided and its LOB size is equal or less than the memory threshold value, then LOB data is stored in memory.
- If <memory_threshold_value> is NULL, then all LOB data is stored in memory.
- If <memory_threshold_value> is 0, then all LOB data is stored on disk.

**ddic_data_type**

Specifies a DDIC data type.

```plaintext
<ddic_data_type> ::=   DDIC_ACCP | DDIC_ALNM | DDIC_CHAR | DDIC_CDAY | DDIC_CLNT |
DDIC_CUKY | DDIC_CURR | DDIC_D16D |
| DDIC_D34D | DDIC_D16R | DDIC_D34R | DDIC_D16S | DDIC_D34S |
DDIC_DATS | DDIC_DAY | DDIC_DEC |
| DDIC_FLTP | DDIC_GUID | DDIC_INT1 | DDIC_INT2 | DDIC_INT4 |
DDIC_INT8 | DDIC_LANG | DDIC_LCHR |
| DDIC_MIN | DDIC_MON | DDIC_LRAW | DDIC_NUMC | DDIC_PREC |
DDIC_QUAN | DDIC_RAW | DDIC_RSTR |
| DDIC_SEC | DDIC_SRST | DDIC_SSTR | DDIC_STRG | DDIC_STXT |
DDIC_TIMS | DDIC_UNIT | DDIC_UCLM |
| DDIC_UTCL | DDIC_UTCS | DDIC_TEXT | DDIC_VARC | DDIC_WEEK
```

**default_value_clause**

Specifies a value to be assigned to the column if an INSERT statement does not provide a value for the column.

```plaintext
<default_value_clause> ::= DEFAULT <default_value_exp>  
<default_value_exp> ::=   NULL   | ...    | <datetime_value_function>   | <string_value_function>  
<datetime_value_function> ::=  CURRENT_DATE  
| CURRENT_TIME  
| CURRENT_TIMESTAMP  
| CURRENT_UTCDATE  
| CURRENT_UTCTIME  
| CURRENT_UTCTIMESTAMP  
<string_value_function> ::=  SYSUUID  
| CURRENT_USER  
| CURRENT_SCHEMA  
| SESSION_USER
```

**col_gen_as_expression**
Specifies the expression to generate the column value in runtime.

<col_gen_as_expression> ::= GENERATED ALWAYS AS <expression>

Generated columns are only supported in column store table definitions, and can only reference base columns that are defined prior to them in the DDL statement (that is, to the left of them). You cannot define a default for a generated column, and they cannot be used as part of a primary key or unique constraint. <expression> must not contain a non-deterministic function.

You cannot drop a base column that is referenced by a generated column, and you

<col_gen_as_ident>

Specifies an identity column. Only one identity column is allowed per table

<col_gen_as_ident> ::= GENERATED { ALWAYS | BY DEFAULT } AS IDENTITY
[(<sequence_option>)]
<sequence_option> ::= (  
<sequence_param_list>
| RESET BY <subquery>
)  
<sequence_param_list> ::= <sequence_parameter> [,<sequence_parameter>] [,...]
<sequence_parameter> ::=   START WITH <start_value>   | INCREMENT BY <increment_value>   | MAXVALUE <max_value>   | MINVALUE <min_value>   | CYCLE   | NO CYCLE   | CACHE <cache_size>   | NO CACHE   | RESET BY <subquery>

There are two types of IDENTITY columns: GENERATED ALWAYS, and GENERATED BY DEFAULT:

GENERATED ALWAYS

A GENERATED ALWAYS AS IDENTITY column increments based solely on the sequence parameter settings (<sequence_param_list>). With the exception of changes to the RESET BY definition, the sequence defined for a GENERATED ALWAYS IDENTITY column cannot be altered or reset after it has been set, and inserting a value into a GENERATED ALWAYS identity column fails and returns an error.

GENERATED BY DEFAULT

A GENERATED BY DEFAULT IDENTITY column increments using the sequence parameter settings (<sequence_param_list>) only when no identity value is specified in the INSERT statement (that is, the identity column is left out of the insert column list). When a value is specified for the identity column during the INSERT, then the specified value is inserted (assuming the value does not already exist) and the following behavior occurs with respect to the value of the identity column:

• If the specified value is higher than the existing identity value with a positive increment, then this resets the current sequence value to an allowable sequence value that is equal to the specified value, or the next one lower than the specified value.
• If the specified value is lower than the existing value with a negative increment, then this resets the current sequence value to an allowable sequence value that is equal to the specified value, or the next one higher than the specified value.
• Otherwise, the current sequence value is not reset.

Identity columns are supported in global and local temporary tables, both for row store and column store tables. They can be added to, or dropped from, a table. However, they cannot be altered.

If RESET BY `<subquery>` is specified, then during the restart of the database, the database automatically executes the subquery and restarts the sequence value with the returned value.

If RESET BY is not specified, then the sequence value is stored persistently in database.

During the restart of the database, the next value of the sequence is generated from the saved sequence value.

For more information on subqueries, see the SELECT statement.

For more information on sequences and sequence parameters, see the CREATE SEQUENCE statement.

col_calculated_field

Specifies the expression that replaces the column at query compilation time.

```plaintext`
<col_calculated_field> ::= AS <expression>
```

The data type of a calculated field column can be a default value. Also, calculated field columns can reference base columns and other calculated field columns.

The maximum value of a recursive calculated field is 16.

Dropping or altering the base columns of a calculated field is not possible.

schema_flexibility

Specifies whether the column is dynamic.

```plaintext`
<schema_flexibility> ::= { ENABLE | DISABLE } SCHEMA FLEXIBILITY
```

ENABLE changes the column into a dynamic column. DISABLE (default value) changes the column into a static column.

fuzzy_search_index

Turns a fuzzy search index on or off. OFF is the default.

```plaintext`
<fuzzy_search_index> ::= FUZZY SEARCH INDEX [ON | OFF]
```

fuzzy_search_mode

Sets the fuzzy search mode with the value of `<string_literal>`. If NULL is specified, then the fuzzy search mode is reset.

```plaintext`
<fuzzy_search_mode> ::= FUZZY SEARCH MODE [ <string_literal> | NULL ]
```

column_constraint_short

The description of the syntax for `<column_constraint_short>` and `<column_constraint>` are the same except that `<column_constraint_short>` does not include syntax for
specifying `<unique_specification>`, and is used specifically when adding a column (`<add_columns_clause>`).

```plaintext
<column_constraint_short> ::= NULL
| NOT NULL
| [ <constraint_name_definition> ] <references_specification>
```

**column_constraint**

Specifies constraints on a column.

```plaintext
<column_constraint> ::= NULL
| NOT NULL
| { HIDDEN | NOT HIDDEN }  | [ CONSTRAINT <constraint_name> ] { <unique_specification> | <references_specification> }
```

**NULL**

Allows NULL values in the column. If NULL is specified it is not considered a constraint, it represents that a column that may contain a null value. The default is NULL.

**NOT NULL**

Prohibits NULL values in the column.

**HIDDEN | NOT HIDDEN**

Specifies whether to hide the column; if not specified, the default behavior is NOT HIDDEN. A hidden column is excluded from a SELECT * operation on a table. It is also excluded in an INSERT INTO, VALUES operation unless the column list specifically references it. A hidden column still appears in system views such as TABLE_COLUMNS, INDEX_VIEWS, PARTITIONS, and so on. In the TABLE_COLUMNS system view, the IS_HIDDEN column indicates whether a column is hidden. Hidden columns still appear when using smart data access and in federation with other databases.

**unique_specification**

Specifies a uniqueness constraint on a column. There are two uniqueness constraints you can set: UNIQUE and PRIMARY KEY.

```plaintext
<unique_specification> ::= UNIQUE [ <rs_tree_type_index> | <cs_inverted_type_index> ]
| PRIMARY KEY [ <rs_tree_type_index> | <cs_inverted_type_index> ]
```

When the optional index type (`<rs_tree_type_index>`, `<cs_inverted_type_index>`) is omitted, the database server chooses the appropriate type by considering the table type and the column data type.

`rs_` and `cs_` at the beginning of the specification types indicate which type of table the specification corresponds to (row store and column store, respectively). If you mistakenly specify a row store index type and the table is a column store table, then the specification is ignored and the index is created as INVERTED VALUE (default). If you mistakenly specify a column store index type and the table is a row store table, an error is returned.

For column-store tables, you can control the default index type behavior using configuration settings in `indexserver.ini`. Specifically, `cs_composite_primary_key_constraint_index_type` controls the default for multi-column
primary keys, while cs_composite_unique_constraint_index_type controls the default for unique indexes. See the SAP HANA Configuration Parameter Reference for more information.

**UNIQUE**

Specifies that the index must have unique values. The optional `<rs_tree_type_index>` specification is only applicable to row store tables.

A composite unique key enables the specification of multiple columns as a unique key. With a unique constraint, multiple rows cannot have the same value in the same column.

A UNIQUE column can contain multiple NULL values.

**PRIMARY KEY**

Specifies a primary key constraint, which is a combination of a NOT NULL constraint and a UNIQUE constraint.

**rs_tree_type_index**

Specifies the tree type for the index. Tree type specification is only applicable to row store tables.

```plaintext
<rs_tree_type_index> ::= { BTREE | CPBTREE }
```

**CPBTREE**

Specifies a CPB+-tree index for row store tables. CPB+-tree stands for Compressed Prefix B+-Tree, which is based on pkB-tree. CPB+-tree is a very small index because it uses 'partial key' that is only part of full key in index nodes. CPB+-tree shows better performance than B+-Tree for larger keys. Specify this type of tree for the following scenarios:

- for character string types
- for binary string types
- for decimal types
- when the constraint is a composite key
- when the constraint is a non-unique constraint

**BTREE**

Specifies a B+-tree index. B+-tree indexes maintain sorted data, which performs efficient insertion, deletion, and search of records. Specify this type of tree for scenarios not described for CPBTREE.

**cs_inverted_type_index**

Specifies the inverted index type for column store tables. Inverted index types are only applicable to column store tables.

```plaintext
<cs_inverted_type_index> ::= INVERTED { HASH | VALUE | INDIVIDUAL }
```

**INVERTED HASH**

INVERTED HASH should not be used as a composite type in cases where range queries or similar queries on the composite keys are a significant part of the workload. In these cases, use INVERTED VALUE instead.
INVERTED HASH indexes are deprecated. Only UNIQUE INVERTED HASH indexes are supported.

INVERTED VALUE

INVERTED VALUE is the default index type for column store tables.

INVERTED INDIVIDUAL

An INVERTED INDIVIDUAL index is a lightweight index type for column store tables with reduced memory footprint. The name INVERTED INDIVIDUAL reflects that internally only inverted indexes on individual columns are created (that is, no concat attributes are created). This type of index is only available for multi-column unique constraints, which may be defined as secondary unique index or primary key.

For <references_specification>, see References Specification [page 601].

add_load_unit

When adding a column to the specified table, specifies how to load data into memory when the table is queried. Specifying the load unit is only supported on column-store tables.

COLUMN LOADABLE

In-memory loading - the entire column is loaded into memory. COLUMN LOADABLE boosts performance at the cost of higher memory usage. This is the default behavior unless another value is inherited.

PAGE LOADABLE

In-buffer cache loading - column data is loaded by page into the buffer cache. PAGE LOADABLE reduces memory usage for specific columns by not requiring those columns to be fully memory resident.

alter_columns_clause

Alters one or more column definitions.

ONLINE [ PREFERRED ]

Allows alter column operations that do not serialize with concurrent DML operations. ONLINE is only supported for column-store, non-replicated tables when altering a column to a LOB type without adding a NOT NULL constraint, or when altering only the default value and/or the nullability of a column. Altering a column in a supported case results in an alter column operation that only acquires a shared table lock. The operation is not serialized with concurrent
DML operations, but is serialized with any other concurrent DDL on the table, including other ONLINE DDL. The default value must be a constant value and not a function.

When PREFERRED is specified, if ONLINE execution is not supported, then a SQL warning message appears, and the alter operation executes in table x-lock mode.

**alter_load_unit**

When altering a column definition, specifies how to load data into memory when the table is queried. Specifying the load unit is only supported on column-store tables.

```plaintext
<alter_load_unit> ::= \{ COLUMN | PAGE | DEFAULT \} LOADABLE
```

**COLUMN LOADABLE**

In-memory loading - the entire column is loaded into memory. COLUMN LOADABLE boosts performance at the cost of higher memory usage. This is the default behavior unless another value is inherited.

**PAGE LOADABLE**

In-buffer cache loading - column data is loaded by page into the buffer cache. PAGE LOADABLE reduces memory usage for specific columns by not requiring those columns to be fully memory resident.

**DEFAULT LOADABLE**

If you specify DEFAULT, then there is no explicit preference for the loading unit that is used when the results are loaded from the table. If there is an inherited loading unit from another object, then that loading unit is used.

**drop_columns_clause**

Removes one or more columns from the specified table.

```plaintext
<drop_columns_clause> ::= DROP ( <column_name> [, ...] ) [ ONLINE [ PREFERRED ] ]
```

**ONLINE [ PREFERRED ]**

Allows drop column operations that do not serialize with concurrent DML operations. ONLINE is only supported for column-store, non-replicated tables when dropping a non-paged column to a LOB type.

When PREFERRED is specified, if ONLINE execution is not supported, then a SQL warning message appears, and the drop operation executes in table x-lock mode.

**with_annotation_clause**

Alters table-, column-, and parameter-level annotations. You can reference annotations in subsequent queries to filter results.

```plaintext
<with_annotation_clause> ::= WITH ANNOTATIONS ( \{ \{ <alter_table_annotations> \} \{ <key_set_operation> | <key_unset_operation> \} \} [ ] <alter_column_annotations> ::= \{ <column_annotation> [ , ... ] \} <alter_table_annotations> ::= \{ <column_annotation> [ , ... ] \} <alter_column_annotations> ::= \{ <column_annotation> [ , ... ] \} <alter_parameter_annotations> ::= \{ <parameter_annotation> [ , ... ] \}
```
While you must specify annotation on at least one object with the WITH ANNOTATION clause (table, column, or parameter), there is no limit on the number of annotations or types of annotations you can specify.

<key> and <value> represent the annotations you are configuring for the object (table, column, or parameter).

system_versioning_configuration

Alters system-versioning settings for a table.

valid_from_column_name

In both <history_table_definition> and <system_versioned_table_definition>, this column defines the column that holds the start time of the validity period. Defining a column as a ROW START column initializes the column with the timestamp of the start of the current transaction, and GENERATED ALWAYS is the mechanism by which the system automatically updates the TIMESTAMP value in the column.

valid_to_column_name
In both `<history_table_definition>` and `<system_versioned_table_definition>`, this column defines the column that holds the end time of the validity period. Defining a column as a ROW END column initializes the column with the timestamp value ‘9999-12-31 23:59:59.9999999’.

**ADD PERIOD FOR SYSTEM_TIME ( valid_from_column_name, valid_to_column_name ) | DROP PERIOD FOR SYSTEM_TIME**

Adds or drops a validity period.

For the ADD syntax, `<valid_from_column_name>` is the column containing the start time value, and `<valid_to_column_name>` is the column containing the end time value.

**ADD SYSTEM VERSIONING HISTORY TABLE history_table_name | DROP SYSTEM VERSIONING**

Enables or disable system-versioning for the table.

For the ADD syntax, `<history_table_name>` must already exist and not be associated with any other system-versioned table.

[[ NOT ] VALIDATED ]

Specifies whether to check if `<history_table_name>` is empty. If VALIDATED is specified (the default), an exception is raised if `<history_table_name>` is not empty. If NOT VALIDATED is specified, no emptiness check is performed on `<history_table_name>`, and no exception is raised even if the history table has data in it.

After you import a system-versioned table and its associated history table, you execute an ALTER TABLE statement to enable system-versioning. Since the history table is not empty, you must specify NOT VALIDATED to bypass the emptiness check.

**add_drop_application_time_period**

Defines the application-time period for the table, or drops the existing one.

```
ADD PERIOD FOR APPLICATION_TIME ( <validfrom_column_name>,
  <validto_column_name> )  | DROP PERIOD FOR APPLICATION_TIME
```

Adding a period adds an implicit constraint that `<validfrom_column_name>` must be less than `<validto_column_name>`.

**lob_reorganize_clause**

Applies the midsizelob_threshold system property to the specified LOB column or to all LOB columns in the specified table.

```
<lob_reorganize_clause> ::=    LOB REORGANIZE [(<column-list>)] [ ONLINE [ PREFERRED ] ]
```

If a comma-separated list is not provided, then the midsizelob_threshold system property is applied to any LOB column in the specified table.

**ONLINE**

Allows LOB reorganize operations that do not serialize with concurrent DML operations. ONLINE is only supported for column-store, non-multistore tables, where specifying ONLINE performs a LOB reorganize operation that only acquires a shared table lock. The operation is still serialized with other concurrent DDL operations on the table, including other ONLINE DDL operations

**PREFERRED** When specified with the ONLINE option, if ONLINE execution is not supported, then a SQL warning message appears, and the add operation executes in table x-lock mode.

**clear_column_join_data_statistics_clause**
Clears the join data statistics for the specified table.

```sql
<clear_column_join_data_statistics_clause> ::= CLEAR COLUMN JOIN DATA STATISTICS
```

**constraint_clauses**

Defines the constraint operations you can perform.

```sql
[<add_constraint_clause>]
[<drop_constraint_clause>]
[<alter_constraint_enforcement_clause>]
```

**add_constraint_clause**

Adds a table constraint.

```sql
<add_constraint_clause> ::= ADD [CONSTRAINT <constraint_name>] <table_constraint>
<constraint_name> ::= <identifier>
```

**table_constraint**

Specifies either a unique constraint, a referential constraint, or a check constraint.

```sql
<table_constraint> ::= <unique_constraint_definition> | <referential_constraint_definition> | <check_constraint_definition>
```

**unique_constraint_definition**

Specifies a unique constraint.

```sql
<unique_constraint_definition> ::= <unique_specification> (<unique_column_name_list>)
```

For more information about unique constraints, see the CREATE TABLE statement.

**unique_column_name_list**

Specifies the unique column name list, which can have one or more column names.

```sql
<unique_column_name_list> ::= <unique_column_name>[, <unique_column_name>]...]
<unique_column_name> ::= <identifier>
```

**referential_constraint_definition**

Specifies a referential constraint.

```sql
<referential_constraint_definition> ::= FOREIGN KEY (<referencing_column_name_list>) <references_specification>
```

To ensure uniqueness, the target of a foreign key constraint must be either a full primary key or a unique column. Self-referencing foreign key constraints are supported.

**referencing_column_name_list**
Specifies the target column(s) of the foreign key constraint.

```
<referencing_column_name_list> ::= <referencing_column_name> [ { ,
<referencing_column_name> } ]...
```

**referencing_column_name**

The identifier of a referencing column.

```
<referencing_column_name> ::= <identifier>
```

**references_specification**

Specifies the referenced table, with optional column name list and trigger action.

```
<references_specification> ::= REFERENCES <referenced_table> [ ( <referenced_column_name_list> ) ]
[ <constraint_enforcement> ]
[ <constraint_check_time> ]
<constraint_enforcement> ::= <enforcement_option>
| <validation_option>
| <constraint_check_time> ::= INITIALLY { IMMEDIATE | DEFERRED }
```

Specifying NOT VALIDATED means that existing data will not be checked to determine whether it violates the foreign key constraint.

Specifying NOT ENFORCED means that new or modified rows will not be checked to determine whether they violate the foreign key constraint.

Enable or disable constraint enforcement rather than dropping and recreating the constraint.

The default `<enforcement_option>` is ENFORCED and the default `<validation_option>` is VALIDATED.

**referenced_table**

Specifies the identifier of a table to be referenced.

```
<referenced_table> ::= <identifier>
```

**referenced_column_name_list**

Specifies the referenced column name list, which can have one or more column names.

```
<referenced_column_name_list> ::= <referenced_column_name> [,,
<referencing_column_name> [,..] ]
```

<referenced_column_name> specifies the identifier of the column name to be referenced.

If `<referenced_column_name_list>` is specified, then there is a one-to-one correspondence between `<column_name>` of `<column_definition>` (see column_definition [page 590]) and `<referenced_column_name>`. If it is not specified, then there is a one-to-one correspondence between `<column_name>` of `<column_definition>` and the column name of the referenced table’s primary key.
**referential_triggered_action**

Specifies an update rule with optional delete rule or a delete rule with optional update rule.

```
<referential_triggered_action> ::= <update_rule> [ <delete_rule> ]
| <delete_rule> [ <update_rule> ]
```

The order that you specify the rules in provides an order of precedence for execution.

**update_rule**

Specifies the behavior to perform when data in the column is updated.

```
<update_rule> ::= ON UPDATE <referential_action>
<referential_action> ::= { RESTRICT | CASCADE | SET NULL | SET DEFAULT }
```

<table>
<thead>
<tr>
<th>Action Name</th>
<th>Update Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTRICT</td>
<td>Any updates to a referenced table are prohibited if there are any matched records in the referencing table. This is the default action.</td>
</tr>
<tr>
<td>CASCADE</td>
<td>If a record is updated in the referenced table, then the corresponding records in the referencing table are also updated with the same values.</td>
</tr>
<tr>
<td>SET NULL</td>
<td>If a record is updated in the referenced table, then the corresponding records in the referencing table are also updated with null values.</td>
</tr>
<tr>
<td>SET DEFAULT</td>
<td>If a record is updated in the referenced table, then the corresponding records in the referencing table are also updated with their default values.</td>
</tr>
</tbody>
</table>

**delete_rule**

Specifies the behavior to perform when data in the column is deleted.

```
<delete_rule> ::= ON DELETE <referential_action>
```

The following DELETE referential actions are possible:

<table>
<thead>
<tr>
<th>Action Name</th>
<th>Delete Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTRICT</td>
<td>Any deletions to a referenced table are prohibited if there are any matched records in the referencing table. This is the default action.</td>
</tr>
<tr>
<td>Action Name</td>
<td>Delete Action</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>CASCADE</td>
<td>If a record in the referenced table is deleted, then the corresponding records in the referencing table are also deleted.</td>
</tr>
<tr>
<td>SET NULL</td>
<td>If a record in the referenced table is deleted, then the corresponding records in the referencing table are set to null.</td>
</tr>
<tr>
<td>SET DEFAULT</td>
<td>If a record in the referenced table is deleted, then the corresponding records in the referencing table are set to their default values.</td>
</tr>
</tbody>
</table>

**constraint_check_time**

Specifies when to check constraints. The default is INITIALLY IMMEDIATE.

- **IMMEDIATE** - referential constraints are checked immediately during statement execution. If a referential constraint is violated, then the statement fails.
- **DEFERRED** - referential constraints are checked at commit time. If a referential constraint is violated, then the transaction is rolled back. Also, if `<referential_triggered_action>` is set to something other than RESTRICT, then the referential constraint check on the parent (referencing) table is not deferred and instead is checked immediately.

**check_constraint_definition**

Specifies a condition to check for in the column.

```sql
<check_constraint_definition> ::= CHECK ( <search_condition> )
```

A table check constraint is satisfied if `<search_condition>` evaluates to true.

**drop_constraint_clause**

Drops a unique or referential constraint.

```sql
<drop_constraint_clause> ::= DROP CONSTRAINT <constraint_name>
```

**alter_constraint_enforcement_clause**

Alters enforcement for a table constraint.

```sql
<alter_constraint_enforcement_clause> ::= ALTER CONSTRAINT <constraint_name> <constraint_enforcement>
<constraint_enforcement> ::= <enforcement_option> | <validation_option> | <enforcement_option> <validation_option>
<enforcement_option> ::= [NOT] ENFORCED
<validation_option> ::= [NOT] VALIDATED
```

When you specify NOT VALIDATED, existing data is not checked to determine whether it violates the foreign key constraint.
When you specify NOT ENFORCED, means that new or modified rows are not checked to determine whether they violate the foreign key constraint.

Enable or disable constraint enforcement rather than dropping and recreating the constraint.

The default `<enforcement_option>` is ENFORCED and the default `<validation_option>` is VALIDATED.

**convert_index_type**

Converts a unique index, unique table constraint, or a primary key on a column store table to an INVERTED index type.

```
ALTER CONSTRAINT `<constraint_name>` UNIQUE `<cs_inverted_type_index>`  
| ALTER PRIMARY KEY `<cs_inverted_type_index>`
```

**cs_inverted_type_index**

Specifies the inverted index type for column store tables. Inverted index types are only applicable to column store tables.

```
<cs_inverted_type_index> ::= INVERTED { HASH | VALUE | INDIVIDUAL }
```

**INVERTED HASH**

INVERTED HASH should not be used as a composite type in cases where range queries or similar queries on the composite keys are a significant part of the workload. In these cases, use INVERTED VALUE instead.

**Note**

Non-unique INVERTED HASH indexes are deprecated. Only UNIQUE INVERTED HASH indexes are supported.

**INVERTED VALUE**

INVERTED VALUE is the default index type for column store tables.

**INVERTED INDIVIDUAL**

An INVERTED INDIVIDUAL index is a lightweight index type for column store tables with reduced memory footprint. The name INVERTED INDIVIDUAL reflects that internally only inverted indexes on individual columns are created (that is, no concat attributes are created). This type of index is only available for multi-column unique constraints, which may be defined as secondary unique index or primary key.

**clientside_encryption_clauses**

Defines the alter operations you can perform with regards to client-side encryption.

**cancel_continue_clientside_encryption**

Restarts or cancels (rolls back) an encryption/decryption/re-encryption operation.

```
<cancel_continue_clientside_encryption> ::=  
CONTINUE CLIENTSIDE ENCRYPTION  
| CANCEL CLIENTSIDE ENCRYPTION
```

**CONTINUE CLIENTSIDE ENCRYPTION**
Restarts a stopped encryption/decryption/re-encryption operation, avoiding the need to redo the operation from the beginning.

Use the CLIENTSIDE_ENCRYPTION_STATUS column of the TABLE_COLUMNS system view to determine the status of an encryption/decryption/re-encryption operation.

**CANCEL CLIENTSIDE ENCRYPTION**

Rolls back the data to the state it was in before the encryption/decryption/re-encryption operation started.

Only the session owner or an administrator can interrupt (stop) a session that is performing an encryption/decryption/re-encryption operation. After the operation has been interrupted, either this command or a CONTINUE command must be executed.

Use the CLIENTSIDE_ENCRYPTION_STATUS column of the TABLE_COLUMNS system view to determine the status of an encryption/decryption/re-encryption operation.

**activate_clientside_encryption**

Applies the new version of the encryption column key to the table where the data is re-encrypted with the new version of the encryption column key.

```sql
ALTER ( <column_name> ALTER CLIENTSIDE ENCRYPTION ACTIVATE NEW VERSION )
```

Only one column at a time can be altered to activate the new version of the column encryption key on the encrypted column.

**enable_disable_clientside_encryption**

Enables client-side encryption on a column using a specified column encryption key (CEK), or disables client-side encryption on a column. Only one column at a time can be altered to enable or disable client-side encryption.

```sql
<enable_disable_clientside_encryption> ::=    ALTER ( <column_name> ALTER CLIENTSIDE ENCRYPTION { OFF | ( ON WITH <column_encryption_key_name> [ RANDOM | DETERMINISTIC ] ) } ) <column_encryption_key_name> ::= [ <schema_name>.]<identifier>
```

*<column_encryption_key_name>* must be a valid column encryption key. Also, *<column_encryption_key_name>* and the table being altered must exist in the same schema.

**RANDOM** and **DETERMINISTIC** describe the type of encryption used by the algorithm to encrypt the data. **RANDOM** ensures that a given cleartext value is never encrypted into the same ciphertext value more than once. **DETERMINISTIC** means that a given cleartext value is always encrypted into the same ciphertext value. Deterministic encryption allows some operations such as equality comparisons on encrypted data, whereas non-deterministic encryption (RANDOM), while stronger, limits operations to basic inserts, updates, and fetches. Choose the type of encryption based on how the data is intended to be used. RANDOM is the default setting.

**OFF** disables client-side encryption on the column and decrypts the column data.

**rotate_clientside_encryption_key**

Re-encrypts the column data using a different column encryption key. Only one column at a time can be altered to rotate the column encryption key on the encrypted column.

```sql
ALTER ( <column_name> ALTER CLIENTSIDE ENCRYPTION WITH <new_column_encryption_key_name> [ RANDOM | DETERMINISTIC ] ) <new_column_encryption_key_name> ::= [ <schema_name>.]<identifier>
```
RANDOM and DETERMINISTIC describe the behavior used by the encryption algorithm to encrypt the data.

<new_column_encryption_key_name> must be a valid column encryption key, and you must have key copies of the old column encryption key and the new column encryption key when re-encrypting.

**primary_key_clauses**

Defines the primary key operations that you can perform.

```sql
<primary_key_clauses> ::=  
  <add_primary_key_clause>  
  | <drop_primary_key_clause>  
  | <primary_key_update_clause>
```

**add_primary_key_clause**

Adds a primary key on a table.

```sql
<add_primary_key_clause> ::=  
  ADD PRIMARY KEY [ <rs_tree_type_index> | <cs_inverted_type_index> ] (  
    <column_name> [, <column_name> [,...] ] )  
  | ADD PRIMARY KEY USING INDEX <index_name>
```

**rs_tree_type_index**

Specifies the tree type for the index. Tree type specification is only applicable to row store tables.

```sql
<rs_tree_type_index> ::= { BTREE | CPBTREE }
```

**CPBTREE**

Specifies a CPB+-tree index for row store tables. CPB+-tree stands for Compressed Prefix B+-Tree, which is based on pkB-tree. CPB+-tree is a very small index because it uses 'partial key' that is only part of full key in index nodes. CPB+-tree shows better performance than B+-Tree for larger keys. Specify this type of tree for the following scenarios:

- for character string types
- for binary string types
- for decimal types
- when the constraint is a composite key
- when the constraint is a non-unique constraint

**BTREE**

Specifies a B+-tree index. B+-tree indexes maintain sorted data, which performs efficient insertion, deletion, and search of records. Specify this type of tree for scenarios not described for CPBTREE.

**cs_inverted_type_index**

Specifies the inverted index type for column store tables. Inverted index types are only applicable to column store tables.

```sql
<cs_inverted_type_index> ::= INVERTED { HASH | VALUE | INDIVIDUAL }
```

**INVERTED HASH**
INVERTED HASH should not be used as a composite type in cases where range queries or similar queries on the composite keys are a significant part of the workload. In these cases, use INVERTED VALUE instead.

**Note**

Non-unique INVERTED HASH indexes are deprecated. Only UNIQUE INVERTED HASH indexes are supported.

**INVERTED VALUE**

INVERTED VALUE is the default index type for column store tables.

**INVERTED INDIVIDUAL**

An INVERTED INDIVIDUAL index is a lightweight index type for column store tables with reduced memory footprint. The name INVERTED INDIVIDUAL reflects that internally only inverted indexes on individual columns are created (that is, no concat attributes are created). This type of index is only available for multi-column unique constraints, which may be defined as secondary unique index or primary key.

**USING INDEX index_name**  Specifies a unique index to promote to the primary key. All columns of the index must have the NOT NULL constraint. The table must not use table replication.

### drop_primary_key_clause

Drops the primary key.

```
<drop_primary_key_clause> ::= DROP PRIMARY KEY
```

### primary_key_update_clause

Specifies if UPDATE statements are allowed on primary key columns. This clause is only supported on tables with heterogeneous partitioning.

```
<primary_key_update_clause> ::= PRIMARY KEY UPDATE { ON | OFF };
```

**PRIMARY KEY UPDATE { ON | OFF }**

Specifies if UPDATE statements are allowed on primary key columns. If not specified, then the default is ON.

### preload_clause

Sets or removes the preload flag of the given tables or columns.

```
<preload_clause> ::= PRELOAD ALL
                  | PRELOAD ( <column_name> )
                  | PRELOAD NONE
```

When the preload flag is set, tables are automatically loaded into memory after an index server start. This setting is available for column store tables.

The current status of the preload flag is visible in the system table **TABLES** in the IS_PRELOAD and IS_PARTIAL_PRELOAD columns and in the system table **TABLE_COLUMNS** in the PRELOAD column. Possible values are TRUE, FALSE, and NULL. Preload information can be NULL in the case of a row store, virtual columns, or tables on extended storage.

Using **PRELOAD** with **ALL** sets preload flags of all columns in the table.
PRELOAD (<column_name>) sets the flags of the specified column.

PRELOAD NONE removes the preload flag from all columns.

**table_conversion_clause**

Converts the table storage from ROW to COLUMN or from COLUMN to ROW.

```sql
```

**THREADS number_of_threads**

Specifies how many parallel execution threads should be used for the table conversion.

```sql
THREADS <number_of_threads>
<number_of_threads> ::= <unsigned_integer>
```

The optimal value for the number of threads is the number of available CPU cores. If THREADS is not provided, then the default value of the number of CPU cores specified in the `indexserver.ini` file is used.

**BATCH batch_size**

Specifies the number of rows to be inserted in a batch.

```sql
BATCH <batch_size>
=batch_size> ::= <unsigned_integer>
```

If BATCH is not specified, then the default value of 2,000,000 is used. Inserts into column tables are immediately committed after every <batch_size> records have been inserted. You can only use the BATCH option when a table is converted from ROW to COLUMN storage.

**association_clauses**

Defines the association operations you can perform.

```sql
[<add_association_clause>] [ <drop_association_clause>]  
```

**add_association_clause**

Adds an association clause. Associations are checked at runtime.

```sql
<add_association_clause> ::= ADD ASSOCIATION ( <association_def> )  
<association_def> ::= [ <join_cardinality> ] JOIN <table_name> [ AS <identifier> ] ON <predicate> WITH DEFAULT FILTER <predicate>  
<join_cardinality> ::=  MANY TO ONE  
| MANY TO MANY  
| ONE TO ONE  
| ONE TO MANY  
<table_name> ::= <identifier>
```

The WITH DEFAULT FILTER clause specifies a default predicate to filter column values.

**drop_association_clause**

Drops an association clause.

```sql
<drop_association_clause> ::= DROP ASSOCIATION <identifier>
```
For clauses to alter heterogeneous and non-heterogeneous partitions, see the topic Non-heterogeneous Partition Clauses or Heterogeneous Partition Clauses in this guide.

- Non-heterogeneous Alter Partition Clauses.
- Heterogeneous Alter Partition Clauses.

**table_load_unit_clause**

Specifies how to load data into memory when the table is queried. Specifying the load unit is only supported on column-store tables. `<table_load_unit_clause>` can be set at column, table, and partition levels.

```
<table_load_unit_clause> ::= { COLUMN | PAGE | DEFAULT } LOADABLE [ CASCADE ]
```

In the case of competing or unspecified `<table_load_unit_clause>` settings, the logic described in CREATE TABLE for `<load_unit>` is applied.

**COLUMN LOADABLE**

In-memory loading - the entire column is loaded into memory. COLUMN LOADABLE boosts performance at the cost of higher memory usage. This is the default behavior unless another value is inherited.

**PAGE LOADABLE**

In-buffer cache loading - column data is loaded by page into the buffer cache. PAGE LOADABLE reduces memory usage for specific columns by not requiring those columns to be fully memory resident.

**DEFAULT LOADABLE**

If you specify DEFAULT, then there is no explicit preference for the paging attribute that is used when the results are loaded from the table. If there is an inherited loading unit from another object, then that loading unit is used.

**CASCADE**

Specify CASCADE when you want the paging attribute to apply to all objects below the specified object in the object hierarchy, so table paging attributes apply to partitions.

**alter_persistent_memory_spec_clause**

Alters the persistent memory storage preference at the table, partition, or column level, depending on where the clause is situated in the ALTER statement. For example, when specified inside `<column_definition>`, it enables or disables persistent memory for the column. When specified inside `<add_range_partition_clause>`, it enables or disables persistent memory for the new partition.

**i Note**

This clause is not supported for heterogeneous range partitions.

```
<alter_persistent_memory_spec_clause> ::= PERSISTENT MEMORY
<alter_pm_preference> [ IMMEDIATE | DEFERRED ] [ CASCADE ]
<alter_pm_preference> ::= { ON | OFF | DEFAULT }
```

**ON IMMEDIATE**

Enables persistent memory usage and populates the underlying PERSISTENT MEMORY storage.

**ON or ON DEFERRED**
Enables persistent memory usage without populating the underlying PERSISTENT MEMORY storage. PERSISTENT MEMORY storage population happens as part of next load from disk or delta merge. DEFERRED is the default behavior when ON is specified without IMMEDIATE or DEFERRED.

**OFF IMMEDIATE**

Disables persistent memory usage and reloads data into DRAM. Cleanup of underlying persistent memory is handled asynchronously.

**OFF or OFF DEFERRED**

Disables persistent memory usage without switching in memory data to DRAM. Data is loaded into DRAM as part of next delta merge or reload. Deletion of persistent memory is driven via delta merge cleanup and subsequent savepoint. DEFERRED is the default behavior when OFF is specified without IMMEDIATE or DEFERRED. Cleanup of underlying persistent memory is handled asynchronously.

**DEFAULT**

No explicit preference is set for persistent memory usage at the specified object level (table, partition or column). Instead, the SAP server, falls back to default behavior, which is to use the inherited persistent memory preference, if any.

IMMEDIATE and DEFERRED actions are not supported when DEFAULT is specified.

**CASCADE**

The user-specified persistent memory preference is applied to all objects below the specified object in the object hierarchy. CASCADE is not supported at the partition level, and is meaningless at the column level.

**replica_clauses**

Defines the alter operations you can perform with regards to replicas.

```
<replica_clauses> ::=<add_replica_clause>     | <alter_replica_group_clause>     | <drop_replica_clause>     | <enable_disable_replica_clause>     | <move_replica_clause>     | <set_replica_source_clause>
```

**add_replica_clause**

Adds a replica.

```
<add_replica_clause> ::= ADD [ <sync_or_async> ] [ <column_or_row> ] REPLICA [ { <src_table_column_list> } ] [ <replica_partition> ] [ <set_group_option> ] [ AT <replica_locations> ]
```

**column_or_row**

Replica type is specified by `<column_or_row>`. If it is not specified, then the type is the same as that of table.

```
<column_or_row> ::= { COLUMN | ROW }
```

**sync_or_async**

Specifies whether the replica is immediately activated.

```
<sync_or_async> ::= { SYNCHRONOUS | ASYNCHRONOUS }
```
SYNCHRONOUS replicas are activated immediately. ASYNCHRONOUS replicas are created in an empty state; execute an ALTER SYSTEM ENABLE ALL ASYNCHRONOUS TABLE REPLICAS statement to activate them.

**src_table_column_list**
Specifies a subset of the columns in the source table. A replica that has a subset of the columns from the source table is called a column-wise replica.

```plaintext
<src_table_column_list> ::= <column_name> [, <column_name> [,…] ]
```

Although you can specify the columns in any order, the order of the columns in the replica matches the order of columns in the source table.

If you specify all of the columns from the source table, then the replica that is created is no longer considered a column-wise replica.

**replica_partition**
Specifies the partitions for the replicated table.

```plaintext
<replica_partition> ::= <partition_table_clause>
```

The **<replica_partition>** clause supports all partition types not using time selection. See the definition for **<partition_table_clause>**.

**AT replica_locations**
Adds the replica to the specified locations.

```plaintext
<replica_locations> ::= { <indexserver_host_port> | ( <indexserver_host_port>, … ) }  
<indexserver_host_port> ::= '<host_name>:<port_number>'
```

If **<replica_locations>** is not specified, then the replica table is added based on the table placement rule. If no table placement rule is also defined, then the replica table is added one of the slave nodes that is not a source table location and where there is not already an existing replica of the table. If there are no more replica nodes to add a replica table, then an error is returned.

The former syntax, **AT { ALL LOCATIONS | <num_replicas> LOCATIONS }**, is supported but deprecated. Specify locations using the **AT <replica_locations>** syntax instead.

**alter_replica_group_clause**
Changes the group attributes for a replica.

```plaintext
<alter_replica_group_clause> ::=      ALTER REPLICA SET <set_group_option> [ CASCADE ] [ AT <replica_location> ]     | ALTER REPLICA UNSET GROUP [ CASCADE ] [ AT <replica_location> ]     | ALTER REPLICA UNSET GROUP LEAD [ AT <replica_location> ]
```

**drop_replica_clause**
Drops a replica at the specified location.

```plaintext
<drop_replica_clause> ::= DROP REPLICA AT LOCATIONS <replica_locations>  
<replica_locations> ::= { '<indexserver>:<port>' | ( '<indexserver>:<port>', … ) }  
```
**enable_disable_replica_clause**

Enables or disables the replication operation for the specified table.

```
<enable_disable_replica_clause> ::=  
  <source_schema_name>.<source_table_name> { ENABLE | DISABLE }  
  [ SYNCHRONOUS | ASYNCHRONOUS ] REPLICA  
  [ PARTITION <partition_number> ]  
  <source_schema_name> ::= <identifier>  
  <source_table_name> ::= <identifier>  
```

If [SYNCHRONOUS | ASYNCHRONOUS] is not specified, the default is SYNCHRONOUS.

The optional PARTITION <partition_number> syntax enables or disables the overall asynchronous replication for the specified table partition. <partition_number> is allowed only for an actual partition, not for a historical partition.

SYNCHRONOUS is not supported for remote table replication.

**move_replica_clause**

Moves replicas from one index server to another.

```
<move_replica_clause> ::=  
  MOVE REPLICA [ PARTITION <partition_number> ] FROM [ LOCATION ] <from_location> TO [ LOCATION ] <to_location>  
  <from_location> ::= <indexserver_host_port>  
  <to_location> ::= <indexserver_host_port>  
  <indexserver_host_port> ::= '<host_name>:<port_number>'  
```

**set_replica_source_clause**

Swaps the source and replica for replication, so that the replica becomes the source and the source becomes the replica.

This clause is only supported if the columns in the source table and replica are the same.

```
<set_replica_source_clause> ::= SET REPLICA SOURCE AT  
  '<host_name>:<port_number>'  
```

**redo_log_option**

Enables or disables the redo log creation for the table.

```
<redo_log_option> ::= ( ENABLE | DISABLE ) DELTA LOG  
```

The ENABLE DELTA LOG statement can only be executed if no system restart has occurred between the DISABLE and ENABLE commands.

After enabling the feature, it is essential to execute a savepoint to ensure that all data is securely stored. Additionally, a data backup must be performed, as data recovery will be impossible without it. If logging is disabled, log entries for this table will not be retained. Changes to the table will only be written to the data store upon performing a savepoint. This may result in a loss of committed transactions if the index server unexpectedly terminates. In such an event, the table must be truncated, and all data must be reinserted. It is advisable to use this option only during the initial load.

Be aware that this clause can cause data discrepancies between primary and secondary sites, so exercise caution when using it.
**auto_merge_option**

Enables or disables automatic delta merge on the specified table.

```plaintext
<auto_merge_option> ::= {ENABLE | DISABLE} AUTOMERGE
```

The delta merge feature is only supported on column store tables.

**unload_priority**

Sets the priority of table to be unloaded from memory.

```plaintext
<unload_priority_option> ::= UNLOAD PRIORITY <unload_priority>
<unload_priority> ::= <digit>
```

<unload_priority> can be 0 ~ 9, where 0 means not-unloadable and 9 means earliest unload.

**schema_flexibility_option**

Enables, disables, or alters schema flexibility for the specified table.

```plaintext
<schema_flexibility_option> ::=   { {ENABLE | DISABLE} SCHEMA FLEXIBILITY}   | ALTER SCHEMA FLEXIBILITY <flexibility_option> [ <flexibility_option> [...] ]
```

**ALTER SCHEMA FLEXIBILITY**

Alters the schema flexibility for the specified table.

```plaintext
ALTER SCHEMA FLEXIBILITY <flexibility_option> [ <flexibility_option> [...] ]
```

All schema flexibility options that are listed in the CREATE TABLE statement in the WITH SCHEMA FLEXIBILITY section can be used here.

**trigger_option**

Enables or disables the specified trigger on the specified table.

```plaintext
<trigger_option> ::= { ENABLE | DISABLE } TRIGGER [<trigger_name>]
```

If `<trigger_name>` is not specified, then this setting enables or disables all triggers on the specified table.

**group_option_clauses**

Defines the group option operations that you can set.

```plaintext
[<set_group_option>]  
[<unset_group_option>]
```

**set_group_option**

Sets one or more table group option for the table. Table group settings are stored in the TABLE_GROUPS system view.

```plaintext
<set_group_options> ::= SET <option> [...]  
<option> ::=    GROUP TYPE <identifier>    | GROUP SUBTYPE <identifier>    | GROUP NAME <identifier>   | GROUP LEAD
```

**unset_group_option**
Unsets (removes) all table group attributes for the table (type, subtype, name, and so on).

```sql
<unset_group_option> ::= UNSET GROUP
```

**row_order_clauses**

Defines the row order clauses you can set.

```sql
[<set_row_order>]
[<unset_row_order>]
```

**set_row_order**

Sets the row order and type.

```sql
<set_row_order> ::= SET <row_order_definition>
<row_order_definition> ::= <row_order_specification>
(rowunique_column_name_list>)
<row_order_specification> ::= ROW ORDER [BY VALUE]
<unique_column_name_list> ::= <unique_column_name> [,…]
```

Rows are sorted exactly by value for a given set of columns. BY VALUE is the default.

**unset_row_order**

Unsets the row order and type.

```sql
<unset_row_order> ::= UNSET ROW ORDER
```

**unused_retention_period**

Unloads a table or table partition from memory if it was not accessed for the number of seconds defined as the retention period.

```sql
<unused_retention_period> ::= UNUSED RETENTION PERIOD { <retention_period> | ( <retention_period> [,…] )}  
<retention_period> ::= <unsigned_integer>
```

A single value specifies the retention period at table-level, including all of the table partitions. Multiple values specify specific retention periods for each of the table partitions. If multiple values are specified, then the number of values must match the number of table partitions.

The default value and behavior of UNUSED_RETENTION_PERIOD is 0 (inactive). To activate UNUSED_RETENTION_PERIOD, change the value to something other than 0. However, use of the UNUSED_RETENTION_PERIOD setting on a table requires that the global unused_retention_period setting in the global.ini configuration file be set to something other than 0.

**reclaim_data_space_clause**

Reclaims data space of the specified table. This clause is only for use on row store tables.

```sql
<reclaim_data_space_clause> ::= RECLAIM DATA SPACE
```

This statement has the same effect as defragmentation by restructuring table data space. However, this behavior does not always guarantee that restructured data space uses less memory than before. In some optimized cases, restructured data space uses much memory.

**series_reorganize_clause**
Reorganizes a series table.

```sql
<series_reorganize_clause> ::= SERIES REORGANIZE [ PART <int_const_partnum> ] [ LIMIT <inst_const_reorganize_limit> ]
```

All table partitions are reorganized by default. If an error occurs during the reorganize operation, then it is returned from the blocking statement.

By default, the statement blocks until all reorganized work is completed.

**PART** \(<\text{int}\_\text{const}\_\text{partnum}\)**

Reorganizes the specified partition.

**LIMIT** \(<\text{inst}\_\text{const}\_\text{reorganize}\_\text{limit}\)**

Limits the number of rows processed by the statement.

**owner_to_clause**

Specifies a new owner for the table.

```sql
<owner_to_clause> ::= OWNER TO <user_name>
```

The new owner becomes the grantor for all of the privileges on the table that were granted by the old owner. Also, the old owner is granted all privileges for the table by the new owner so that they can continue running jobs on the table.

**alter_mask_settings_clause**

Add a new mask expression or drop an existing mask expression. Data masking transforms confidential data so that it appears meaningless to users who don’t have the privileges required to view it. **alter_mask_settings_clause** does not change the table’s column definitions.

```sql
<alter_mask_settings_clause> ::= { ADD | ALTER } [{ DEFAULT | SESSION USER }] MASK ( <columns_name> USING <mask_expression> [, ]... ) | DROP MASK ( <column_name_list> )<mask_expression> ::= <expression> <column_name_list> ::= <column_name>[, ]...
```

When adding a mask expression, the mask behavior is defined for the object. New column masks must use the same mask mode. When altering, you can change the mask expression, but not the mode. Once all existing masks are dropped, you can add new masks using a different mask mode.

Masking behavior is supported on row and column tables, and SQL row and calculation views. It is not supported on any other type of tables (virtual tables, extended tables, temporary tables etc.) and views (Join/Olap/Hierarchy views).

Only one masking behavior, definer owner-based (DEFAULT MASK) or session user based masking (SESSION USER MASK), is supported on a table or view with masked columns. You can combine both masking behaviors in an object hierarchy. For example, a view with session user based masking can be created on a table with owner-based masking.

If not specified, DEFAULT is the default.

**<mask_expression>** can be any type of expression, including a user-defined function, that returns the same data type and length as the original column.
For more information on data masking, see the SAP HANA Security Guide for SAP HANA Platform.

**set_movable_clause**

Controls whether the table can be moved to another location.

```plaintext
<set_movable_clause> ::= SET [NOT] MOVABLE
```

When a table is set to NOT MOVABLE, it cannot be moved to another location. By default, tables are movable.

**numa_node_preference_clause**

Alters the NUMA node preferences. Although this clause is defined here at the table level, `<numa_node_preference_clause>` can be set in various locations such as range partition definitions (not hash or round-robin) and column definitions, as indicated in the syntax within the topic. `<numa_node_preference_clause>` is not supported for heterogeneous partitions.

```plaintext
<numa_node_preference_clause> ::= NUMA NODE { ( <numa_node_index_spec> ) | NULL } [ IMMEDIATE | DEFERRED ]
<numa_node_index_spec> ::= <numa_node_spec> [, <numa_node_spec> [,... ] ]
<single_node_spec> ::= <integer_const>
<range_node_spec> ::= <integer_const> TO <integer_const>
```

**NULL**

To unset existing NUMA node preferences on the associated object (table, column, partition, or replica), specify NUMA NODE NULL.

**IMMEDIATE, DEFERRED**

Specify whether to apply the NUMA node preferences immediately (IMMEDIATE) or to defer applying the preferences until the next time the table is reloaded (DEFERRED). If you specify IMMEDIATE, then the associated table, column, or table partition is immediately moved according to the altered preference settings.

The IMMEDIATE option is not effective if the `<alter_persistent_memory_spec_clause>` (Fast Restart or PMEM) is enabled on this table; in this case the new NUMA node will only be used after the column is rewritten into persistent memory. You can force this by either running a delta merge `<auto_merge_option>`) or explicitly deleting persistent memory so that it is rewritten with the next load: UNLOAD <table> DELETE PERSISTENT MEMORY;

**integer_const**

`<integer_const>` cannot be a negative number.

**numa_node_spec**

Specify one or more single NUMA nodes (`<single_node_spec>`), or one or more NUMA node ranges (`<range_node_spec>`), or a mixture of both.

NUMA node indexes should be specified in the range of 0 to one less than max_numa_node_count, where max_numa_node_count is the number of NUMA nodes configured for the system. If the NUMA node index specified is greater than or equal to max_numa_node_count, then a random NUMA node in the range of 0 to one less than max_numa_node_count is selected for allocation. For example, on a system where max_numa_node_count is set to 8, if you specify a NUMA node index of 10, then any node in the range of 0 to 7 (inclusive) is chosen randomly for allocation.
Description

This statement is not for use with virtual tables. To alter a virtual table, use the ALTER VIRTUAL TABLE statement.

Permissions

In general, you must have the ALTER object privilege on a table to alter it.

Some non-destructive partitioning clauses can be executed with the PARTITION ADMIN system privilege. For all other partitioning clauses, you must have the ALTER object privilege on a table to alter a partition. For details, see Non-heterogeneous Alter Partition Clauses [page 630] and Heterogeneous Alter Partition Clauses [page 640].

For the clauses listed below, only the TABLE ADMIN system privilege is required. These clauses are administrative in nature and do not change the structure of the table and do not allow access to table data either explicitly or implicitly.

- `<lob_reorganize_clause>`
- `<clear_column_join_data_statistics_clause>`
- `<preload_clause>`
- `<table_conversion_clause>`
- `<alter_persistent_memory_spec_clause>`
- `<auto_merge_option>`
- `<unload_priority>`
- `<group_option_clauses>`
- `<row_order_clauses>`
- `<unused_retention_period>`
- `<reclaim_data_space_clause>`
- `<series_reorganize_clause>`
- `<set_movable_clause>`
- `<convert_index_type>`
- `<numa_node_preference_clause>`

Examples

Create Table t, and then alter the default value of column b to 10.

```sql
CREATE ROW TABLE t (a INT, b INT, c INT);
ALTER TABLE t ALTER (b INT DEFAULT 10);
```
Alter table t adding a new column d.

   ALTER TABLE t ADD (d NVARCHAR(10) DEFAULT 'NCHAR');

Drop a primary key on an existing table t.

   ALTER TABLE t DROP PRIMARY KEY;

Add a primary key on table t to columns c and d.

   ALTER TABLE t ADD PRIMARY KEY (c, d);

Add a primary key constraint, prim_key, on columns a and b of existing table t.

   ALTER TABLE t ADD CONSTRAINT prim_key PRIMARY KEY (a, b);

Change the table type of table t to COLUMN storage.

   ALTER TABLE t COLUMN;

Set the preload flags of column b and c on table t.

   ALTER TABLE t PRELOAD (b, c);

Change the unload priority of table t to 2:

   ALTER TABLE t UNLOAD PRIORITY 2;

Apply the midsize LOB threshold system property to the data column of the t table.

   ALTER TABLE t LOB REORGANIZE (a);

Apply the midsize LOB threshold system property to all LOB columns on the t table.

   ALTER TABLE t LOB REORGANIZE;

Create table T3 with page loadable columns C3, C4 and C5. Then, C3 becomes column loadable and C6 page loadable.

   CREATE COLUMN TABLE T3 (C1 INT PRIMARY KEY, C2 DATE, C3 VARCHAR(80) PAGE LOADABLE NOT NULL, C4 VARCHAR(5000) PAGE LOADABLE, C5 VARCHAR(5000) PAGE LOADABLE NOT NULL, C6 VARCHAR(5000));
   ALTER TABLE T3 ALTER (C3 VARCHAR(80) COLUMN LOADABLE, C6 ALTER PAGE LOADABLE);

Set group type and name of T3, and then unset it:

   ALTER TABLE T3 SET GROUP TYPE group_type1 GROUP NAME group_name1;
   ALTER TABLE T3 UNSET GROUP;

Execute the following statements to create and populate table t1 and then perform a drop operation on column a that does not serialize with concurrent DML operations:

   CREATE TABLE t1 (a INT, b INT);
   ALTER TABLE t1 DROP (a) ONLINE;
Alter the primary key index on Table T1 to an inverted HASH type.

```sql
ALTER TABLE T1 ALTER PRIMARY KEY INVERTED HASH;
```

Create table T and unique index IDX, then promote IDX to the primary key of T.

```sql
CREATE COLUMN TABLE T (I INT NOT NULL);
CREATE UNIQUE INDEX IDX ON T (I);
ALTER TABLE T ADD PRIMARY KEY USING INDEX IDX;
```

**JOIN**

Retrieve, display, and delete column join data statistics.

```sql
DROP TABLE A;
CREATE COLUMN TABLE A ( NVARCHAR(2), ab INT);
DROP TABLE B;
CREATE COLUMN TABLE B (b1 INT, b2 NVARCHAR(1));
INSERT INTO A (, ab) VALUES ('', 1);
INSERT INTO A (, ab) VALUES ('BB', 2);
INSERT INTO B (b1, b2) VALUES (1, 'a');
INSERT INTO B (b1, b2) VALUES (3, 'f');
MERGE DELTA OF A;
MERGE DELTA OF B;
```

Perform a join on tables A and B.

```sql
SELECT * FROM A INNER JOIN B ON A.ab = B.b1;
```

View the column join data statistics for tables A and B.

```sql
SELECT COUNT(*) FROM M_JOIN_DATA_STATISTICS WHERE schema_name1 = 'SYSTEM' AND table_name1 IN ('A', 'B');
SELECT * FROM M_JOIN_DATA_STATISTICS;
```

Clear the column join data statistics from the system.

```sql
ALTER SYSTEM CLEAR COLUMN JOIN DATA STATISTICS;
```

Clear the column join data statistics from table A.

```sql
ALTER TABLE A CLEAR COLUMN JOIN DATA STATISTICS;
```

**Schema flexibility**

Create table T1 and then turn on the schema flexibility of the table. You test the schema flexibility of table T1 by inserting a new record. A new column F is automatically created and a record is inserted:

```sql
CREATE COLUMN TABLE T1 (C INT);
ALTER TABLE T1 ENABLE SCHEMA FLEXIBILITY;
INSERT INTO T1 (C, F) VALUES (1, 'NVAR1');
```

**Fuzzy searching**

Create table T2 with a fuzzy search index and fuzzy search mode. You then switch off the fuzzy search index of COL1 and finally reset fuzzy search mode of COL2.

```sql
CREATE COLUMN TABLE T2 (KEY INT, COL1 VARCHAR(10) FUZZY SEARCH INDEX ON, COL2 NVARCHAR(10) FUZZY SEARCH MODE 'postcode');
ALTER TABLE T2 ALTER (COL1 VARCHAR(10) FUZZY SEARCH INDEX OFF);
ALTER TABLE T2 ALTER (COL2 NVARCHAR(10) FUZZY SEARCH MODE NULL);
```
Identity columns

Create table T5, and add an identity column B with starting value of 100 and an increment of 10.

```sql
CREATE COLUMN TABLE T5 (A INT);
ALTER TABLE T5 ADD (B INT GENERATED ALWAYS AS IDENTITY (START WITH 100 INCREMENT BY 10));
```

Create table T_ALWAYS with identity column B that has a start value of 100 and increments by 10.

```sql
CREATE COLUMN TABLE T_ALWAYS (A INT);
ALTER TABLE T_ALWAYS ADD (B INT GENERATED ALWAYS AS IDENTITY (START WITH 100 INCREMENT BY 10));
INSERT INTO T_ALWAYS (A) VALUES (1);
INSERT INTO T_ALWAYS (A) VALUES (2);
SELECT * FROM T_ALWAYS;
```

In the results table below, the Description column isn't part of the results, it is provided to explain the values in column B.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>The START WITH value</td>
</tr>
<tr>
<td>2</td>
<td>110</td>
<td>The next INCREMENT value</td>
</tr>
<tr>
<td>3</td>
<td>105</td>
<td>A specified value that is lower than the current sequence</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
<td>The next INCREMENT value</td>
</tr>
<tr>
<td>5</td>
<td>205</td>
<td>The specified value that resets the internal sequence to 200</td>
</tr>
<tr>
<td>6</td>
<td>210</td>
<td>The next INCREMENT value</td>
</tr>
</tbody>
</table>

Table replication
Create an additional asynchronous table replica for table `SRC_SCHEMA.TAB1`.

```
ALTER TABLE SRC_SCHEMA.TAB1 ADD ASYNCHRONOUS REPLICA;
```

Drop the table replicas of `SRC_SCHEMA.TAB1` in all locations.

```
ALTER TABLE SRC_SCHEMA.TAB1 DROP REPLICA AT ALL LOCATIONS;
```

Drop the table replica of `SRC_SCHEMA.TAB1` on `hostb:30301`.

```
ALTER TABLE SRC_SCHEMA.TAB1 DROP REPLICA AT 'hostb:30301';
```

Enable asynchronous table replication for the table `SRC_SCHEMA.TAB1`.

```
ALTER TABLE SRC_SCHEMA.TAB1 ENABLE ASYNCHRONOUS REPLICA;
```

Disable asynchronous table replication for the table `SRC_SCHEMA.TAB1`.

```
ALTER TABLE SRC_SCHEMA.TAB1 DISABLE ASYNCHRONOUS REPLICA;
```

Create table T4 with replicas in all indexservers. Then, you drop all replicas and create a replica at the specific indexserver. Then, you move the replica to the other indexserver.

```
CREATE ROW TABLE T4 (C1 INT PRIMARY KEY) REPLICA AT ALL LOCATIONS;
ALTER TABLE T4 DROP REPLICA AT ALL LOCATIONS;
ALTER TABLE T4 ADD REPLICA AT 'host_name:30040';
ALTER TABLE T4 MOVE REPLICA FROM 'host_name:30040' TO 'host_name:30042';
```

Add a replica to table `SRC_TABLE`.

```
CREATE COLUMN TABLE SRC_TABLE (C1, C2, C3) PARTITION BY HASH (C1) PARTITIONS 32;
ALTER TABLE SRC_TABLE ADD REPLICA PARTITION BY HASH (C2) PARTITIONS 8;
```

Client-side encryption

The following statements create a table T6 and enable client-side encryption on column `Name` using the `myCEK` column encryption key.

```
CREATE ROW TABLE T6 ( ID INT PRIMARY KEY, Name NVARCHAR(32) );
ALTER TABLE T6 ALTER ( Name ALTER CLIENTSIDE ENCRYPTION ON WITH myCEK RANDOM );
```

The following example changes the column encryption key used to encrypt column `T6.Name` to `myCEK2`, and changes the encryption mode to DETERMINISTIC:

```
ALTER TABLE T6 ALTER ( Name ALTER CLIENTSIDE ENCRYPTION WITH myCEK2 DETERMINISTIC );
```

The following statement cancels the interrupted encryption operation that is taking place on column `T6.Name` and rolls back the column to how it was before the encryption operation began:

```
ALTER TABLE T6 CANCEL CLIENTSIDE ENCRYPTION;
```

The following statement disables column encryption on `T6.Name`:

```
ALTER TABLE T6 ALTER ( Name ALTER CLIENTSIDE ENCRYPTION OFF );
```
The following statement applies a new version of the encryption key to column encryption on T1.C2.

```
ALTER TABLE T1 ALTER ( C2 ALTER CLIENTSIDE ENCRYPTION ACTIVATE NEW VERSION);
```

**Data masking**

Alter the Personal_Information table to use masking for the SSN column:

```
ALTER TABLE PERSONAL_INFORMATION ADD MASK (SSN USING '****');
```

Alter the Personal_Information table to drop masking from the SSN column:

```
ALTER TABLE PERSONAL_INFORMATION DROP MASK (SSN);
```

**Constraints**

Create two tables and then add a foreign key constraint to table REFERENCING_T.

```
CREATE COLUMN TABLE REFERENCED_T ( A INT PRIMARY KEY, B INT);
CREATE COLUMN TABLE REFERENCING_T (A INT, B INT);
ALTER TABLE REFERENCING_T ADD CONSTRAINT fk1 FOREIGN KEY(A) REFERENCES REFERENCED_T(A);
```

Populate the tables REFERENCED_T and REFERENCING_T.

```
INSERT INTO REFERENCED_T VALUES (1, 1);
INSERT INTO REFERENCING_T VALUES (1, 1);
```

Executing the following statement fails with a foreign key constraint violation:

```
INSERT INTO REFERENCING_T VALUES (2, 1);
```

Alter table REFERENCING_T so that the foreign key constraint is not enforced.

```
ALTER TABLE REFERENCING_T ALTER CONSTRAINT fk1 NOT ENFORCED;
```

Now, running the following statement succeeds:

```
INSERT INTO REFERENCING_T VALUES (2, 1);
```

If you then run the following statement, it fails as the constraint is violated by the value that was just inserted:

```
ALTER TABLE REFERENCING_T ALTER CONSTRAINT fk1 ENFORCED;
```

However, if you add NOT VALIDATED to the statement above and re-run it, it succeeds.

```
ALTER TABLE REFERENCING_T ALTER CONSTRAINT fk1 ENFORCED NOT VALIDATED;
```

Running the same INSERT statement as above now fails because the ENFORCED NOT VALIDATED clause means that only existing data is not checked, but modified rows cannot violate the constraint.

```
INSERT INTO REFERENCING_T VALUES (2, 1);
```

Create a table and add a unique constraint UK.

```
CREATE ROW TABLE R (A INT PRIMARY KEY, B NVARCHAR(10));
```
ALTER TABLE R ADD CONSTRAINT UK UNIQUE (B);

Drop the unique constraint UK from table R.

ALTER TABLE R DROP CONSTRAINT UK;

Create table S. You add a referential constraint FK to table S that references column A of table R with delete cascade option.

CREATE TABLE S (FA INT, B NVARCHAR(10));
ALTER TABLE S ADD CONSTRAINT FK FOREIGN KEY(FA) REFERENCES R(A) ON DELETE CASCADE;

The following example creates a column table, creates a history table, and then alters the column table to configure and enable system versioning:

CREATE COLUMN TABLE account (   account_id INT PRIMARY KEY,
   account_owner_id NVARCHAR(10),
   account_balance DOUBLE );
CREATE COLUMN TABLE account_history (   account_id INT,
   account_owner_id NVARCHAR(10),
   account_balance DOUBLE,
   valid_from TIMESTAMP NOT NULL,
   valid_to TIMESTAMP NOT NULL );
ALTER TABLE account ADD (valid_from TIMESTAMP NOT NULL GENERATED ALWAYS AS ROW START);
ALTER TABLE account ADD (valid_to TIMESTAMP NOT NULL GENERATED ALWAYS AS ROW END);
ALTER TABLE account ADD PERIOD FOR SYSTEM_TIME(valid_from, valid_to);
ALTER TABLE account ADD SYSTEM VERSIONING HISTORY TABLE account_history;

Persistent memory

Alter a table’s PERSISTENT MEMORY specification and cascade the setting to partitions and columns:

ALTER TABLE myTable1 PERSISTENT MEMORY ON IMMEDIATE CASCADE;

Alter a column’s PERSISTENT MEMORY specification:

ALTER TABLE myTable2 ALTER (C1 INT PERSISTENT MEMORY DEFAULT);

Add a partition with a PERSISTENT MEMORY specification:

ALTER TABLE myTable3 ADD RANGE (C1) (PARTITION '0' <= VALUES < '10',
PARTITION OTHERS PERSISTENT MEMORY ON DEFERRED);

Partition a table with a PERSISTENT MEMORY specification for selected partitions:

ALTER TABLE myTable4 PARTITION BY RANGE (C1) (PARTITION '0' <= VALUES < '10' PERSISTENT MEMORY ON IMMEDIATE, PARTITION OTHERS);

Alter a partition’s PERSISTENT MEMORY specification:

ALTER TABLE myTable5 ALTER PARTITION 10 PERSISTENT MEMORY DEFAULT;

Non-heterogeneous partitions
Convert a non-partitioned column store table named T1 to a range partitioned table.

```
CREATE COLUMN TABLE T1 (a INT, b INT);
ALTER TABLE T1 PARTITION BY RANGE (a) ( PARTITION 10 <= VALUES < 20);
```

Add a new second-level partition to an existing range-partitioned table, making it a range-range partitioned table.

```
ALTER TABLE T1 PARTITION BY
    RANGE (a) ( PARTITION 10 <= VALUES < 20),
    RANGE (b) ( PARTITION 100 <= VALUES < 150);
```

Redefine the existing first-level partition to ranges 10 - 15 and 15 - 20.

```
ALTER TABLE T1 PARTITION BY
    RANGE (a) ( PARTITION 10 <= VALUES < 15, PARTITION 15 <= VALUES < 20),
    RANGE (b) ( PARTITION 100 <= VALUES < 150);
```

Add a new partition to the second-level partition.

```
ALTER TABLE T1 ADD PARTITION (b) USING DEFAULT STORAGE (150 <= VALUES < 200);
```

Move a partition.

```
ALTER TABLE T1 MOVE PARTITION 1 TO 'myhost:30201';
```

Move multiple partitions to new locations.

```
ALTER TABLE T2 MOVE PARTITION 1,2 TO 'myhost2:34240', PARTITION 3,4 TO 'myhost3:34242' ONLINE;
```

Drop second-level partition range 10 - 15.

```
ALTER TABLE T1 DROP PARTITION (a) 10 <= VALUES < 15;
```

Merge all partitions into a single non-partitioned table.

```
ALTER TABLE T1 MERGE PARTITIONS;
```

Alter column table T1, setting the NUMA node preference at the partition level on NUMA node index 3. Since DEFERRED is specified, it sets only the NUMA preference for the partition; the actual allocations are made when the table is reloaded. IMMEDIATE alters column table T1, setting the NUMA node preference on partition 1, which already exists. The allocations are made immediately, and the table partition is reloaded to node 4.

```
ALTER TABLE T1 ALTER PARTITION 1 NUMA NODE ('4') IMMEDIATE;
```

Create a non-partitioned table named T2, convert the table to a range partitioned table, and then add an additional partition.

```
CREATE COLUMN TABLE T2 (a INT, b INT);
ALTER TABLE T2 PARTITION BY RANGE (a) (PARTITION VALUE = 1, PARTITION OTHERS);
ALTER TABLE T2 ADD PARTITION (a) 2 <= VALUES < 10;
```
Create a non-partitioned table named T3 and then convert the table to a range-range partitioned table.

```
CREATE COLUMN TABLE T3 (a INT, b INT);
ALTER TABLE T3 PARTITION BY RANGE (a) (PARTITION VALUE = 1, PARTITION OTHERS),
  RANGE (b) (PARTITION 2 <= VALUES < 10);
```

Convert an existing hash partitioned table named T4 to a range partitioned table.

```
CREATE TABLE T4 (A INT, B INT) PARTITION BY HASH (A) PARTITIONS 2;
ALTER TABLE T4 PARTITION BY RANGE (A) (PARTITION 10 <= VALUES < 20, PARTITION OTHERS);
```

Convert an existing range partitioned table named T5 to a hash partitioned table.

```
CREATE TABLE T5 (A INT, B INT) PARTITION BY RANGE (A) (PARTITION 10 <= VALUES < 20, PARTITION OTHERS);
ALTER TABLE T5 PARTITION BY HASH (A) PARTITIONS 2;
```

Convert a range-range partitioned (time selection) table named T6 to a range partitioned table (convert RANGE-RANGE to RANGE).

```
CREATE COLUMN TABLE T6 (A INT PRIMARY KEY, B INT, _DATAAGING NVARCHAR(8))
  PARTITION BY RANGE(A) (PARTITION VALUE = 1, PARTITION 10 <= VALUES < 20,
  PARTITION OTHERS), RANGE (_DATAAGING)
  (USING DEFAULT STORAGE (PARTITION value = '00000000' IS CURRENT,
  PARTITION '20100101' <= VALUES < '20110101', PARTITION OTHERS))
  WITH PARTITIONING ON ANY COLUMNS ON
  FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS OFF
  FOR DEFAULT STORAGE NON CURRENT PARTITIONS PAGE LOADABLE;
ALTER TABLE T6 PARTITION BY RANGE(A) (PARTITION VALUE = 1, PARTITION 10 <=
  VALUES < 20, PARTITION OTHERS);
```

**Dynamic Range Partitioning**

These examples are based on table A1 using this CREATE TABLE statement.

```
CREATE COLUMN TABLE A1 (A INT, B INT NOT NULL) PARTITION BY RANGE (A)
  (PARTITION VALUES = 10),
  RANGE (B) (PARTITION VALUES = 20, PARTITION OTHERS);
```

Enable dynamic range partitioning on the OTHERS partition.

```
ALTER TABLE A1 PARTITION OTHERS DYNAMIC;
```

Add another partition from OTHERS.

```
ALTER TABLE A1 ADD PARTITION FROM OTHERS;
```

Drop any empty OTHERS partitions.

```
ALTER TABLE A1 DROP EMPTY PARTITIONS;
```

Change the dynamic threshold to 2, which uses the value defined in the DYNAMIC_RANGE_THRESHOLD field of the TABLE_PLACEMENT table.

```
ALTER TABLE A1 PARTITION OTHERS DYNAMIC THRESHOLD 2;
```
Disable dynamic range partitioning on the OTHERS partition.

```
ALTER TABLE A1 PARTITION OTHERS NO DYNAMIC;
```

**Heterogeneous partitions**

Convert the non-partitioned table T1 to a range-range partitioned table. The table has four first-level partitions (10-20, 20-30, 40, OTHERS). Range 10-20 has three subpartition ranges (0-10, 10-100, others). Range 40 has two subpartitions ranges (0-10, OTHERS). Both subpartitions have dynamic range partitioning enabled.

```
CREATE COLUMN TABLE T1 (a INT, b INT NOT NULL);
ALTER TABLE T1 PARTITION BY RANGE (a)
  ((PARTITION 10 <= VALUES < 20 INSERT OFF, PARTITION 20 <= VALUES < 30 INSERT OFF)
  SUBPARTITION BY RANGE (b) (PARTITION 0 <= VALUES < 10, PARTITION 10 <= VALUES < 100, PARTITION OTHERS DYNAMIC THRESHOLD 2),
  (PARTITION VALUES = 40)
  SUBPARTITION BY RANGE(B) (PARTITION 0 <= VALUES < 10, PARTITION OTHERS DYNAMIC THRESHOLD 2),
  (PARTITION OTHERS));
```

Convert the non-partitioned table T2 to a range-hash partitioned table. The first level has two partition ranges (10-20, 20-30). The INSERT OFF property means you can’t add data to the partitions. The second level is a hash with two partitions.

```
CREATE COLUMN TABLE T2 (a INT, b INT);
ALTER TABLE T2 PARTITION BY RANGE (a)
  ((PARTITION 10 <= VALUES < 20 INSERT OFF, PARTITION 20 <= VALUES < 30 INSERT OFF)
  SUBPARTITION BY HASH (b) PARTITIONS 2);
```

Add a new first-level range partition 41-50 to table T1. The new partition has a subpartition with one range 0-10.

```
ALTER TABLE T1 ADD PARTITION RANGE (a) ((PARTITION 41 <= VALUES < 50 )
  SUBPARTITION by RANGE (b) (PARTITION 0 <= VALUES < 10 ));
```

Redefine the existing first-level partition range 10-20 to ranges 5-10 and 15-20. Each redefined range retains its original subpartition. All existing range values must be accounted for in the redefined structure or they will be dropped if they do not contain data.

```
ALTER TABLE T1 PARTITION BY RANGE (a)
  ((PARTITION 5 <= VALUES < 10, PARTITION 10 <= VALUES < 20 INSERT OFF,
  PARTITION 20 <= VALUES < 30 INSERT OFF)
  SUBPARTITION BY RANGE (b) (PARTITION 0 <= VALUES < 10, PARTITION 10 <= VALUES < 100, PARTITION OTHERS DYNAMIC THRESHOLD 2),
  (PARTITION VALUES = 40) SUBPARTITION BY RANGE(B) (PARTITION 0 <= VALUES < 10, PARTITION OTHERS DYNAMIC THRESHOLD 2),
  (PARTITION 41 <= VALUES < 50 ) SUBPARTITION by RANGE (b) (PARTITION 0 <= VALUES < 10),
  (PARTITION OTHERS));
```

Move partition 1 (range 5-10) on table T1 to host myhost on port 30203.

```
ALTER TABLE T1 MOVE PARTITION 1 TO 'myhost:30203';
```
Move range 5-10 with subpartition range 0 - 10 on table T1 to host myhost on port 30203.

```
ALTER TABLE T1 MOVE RANGE (a) ((PARTITION 5 <= VALUES < 10) SUBPARTITION BY RANGE (b) (PARTITION 0 <= VALUES < 10)) TO 'myhost:30203';
```

Move the hash subpartition of range 10-20 on table T2 to host myhost1 port 30203 and myhost2 port 30203.

```
ALTER TABLE T2 MOVE SUBPARTITIONS OF RANGE (a) ((PARTITION 10 <= VALUES < 20) to ('myhost1:30203', 'myhost2:30203);
```

Drop the first-level partition range 5-10 on table T1. The subpartitions for that range only are dropped. The subpartition ranges 10-20 and 20-30 remain.

```
ALTER TABLE T1 DROP PARTITION RANGE (a) (( PARTITION 10 <= VALUES < 15 ));
```

Drop subpartition range 10-100 in range 15-20 on table T1.

```
ALTER TABLE T1 ALTER PARTITION RANGE (a) ((PARTITION 10 <= VALUES < 20)) DROP PARTITION RANGE (b) ((PARTITION 10 <= VALUES < 100));
```

Assign the group name GROUP1 to partition 2 on table T1.

```
ALTER TABLE T1 ALTER PARTITION 2 SET GROUP NAME "GROUP1";
```

Assign the group name GROUP2 and the group type HR to the partition range 10 - 20 and all its subpartitions on table T1.

```
ALTER TABLE T1 ALTER PARTITION RANGE (a) ((PARTITION 20 <= VALUES < 30)) SET GROUP NAME "GROUP2" GROUP TYPE "HR" CASCADE;
```

Remove the GROUP value from partition 2 on table T1.

```
ALTER TABLE T1 ALTER PARTITION 2 UNSET GROUP;
```

Remove the GROUP value from the partition range 10 -20 and its subpartitions on table T1.

```
ALTER TABLE T1 ALTER PARTITION RANGE (a) ((PARTITION 20 <= VALUES < 30)) UNSET GROUP CASCADE;
```

This example creates a multilevel heterogeneous partitioned table and then applies the `<load_unit>` attribute first by partition number (1) and then to partition range (10 to 20).

```
CREATE COLUMN TABLE T4 (a INT, b INT) PARTITION BY RANGE(a) ( ( PARTITION 10 <= VALUES < 20 ) SUBPARTITION BY RANGE (b) ( PARTITION VALUES = 100 ));
ALTER TABLE T4 ALTER PARTITION 1 COLUMN LOADABLE;
ALTER TABLE T4 ALTER PARTITION RANGE (a) ((PARTITION 10 <= VALUES < 20) SUBPARTITION BY RANGE (b) (PARTITION VALUES = 100)) PAGE LOADABLE;
```

**Dynamic Range Partitioning**

These examples are based on table A1 using this CREATE TABLE statement.

```
CREATE COLUMN TABLE A1 (A INT, B INT NOT NULL) PARTITION BY RANGE (A) ((PARTITION 10 <= VALUES < 20) SUBPARTITION BY RANGE (B) (PARTITION 15 <= VALUES < 20, PARTITION OTHERS),
```
Enable dynamic range partitioning with a DYNAMIC interval of 2 years on range 10-20, subpartition 15-20 on table A1.

```
ALTER TABLE A1 PARTITION BY RANGE (A)
  ((PARTITION 10 <= VALUES < 20)
   SUBPARTITION BY RANGE (B) (PARTITION 15 <= VALUES < 20, PARTITION OTHERS DYNAMIC),
   (PARTITION VALUES = 30)
   SUBPARTITION BY RANGE (B) (PARTITION VALUES = 20, PARTITION OTHERS),
   (PARTITION OTHERS));
```

Add another partition from OTHERS or subpartition 15-20 on table A1.

```
ALTER TABLE A1 ADD PARTITION FROM OTHERS;
```

Drop any empty OTHERS partitions on table A1.

```
ALTER TABLE A1 DROP EMPTY PARTITIONS;
```

Set the threshold to 2 for dynamic range partitioning on the subpartition of range 10-20 on table A1.

```
ALTER TABLE A1 PARTITION BY RANGE (A)
  ((PARTITION 10 <= VALUES < 20)
   SUBPARTITION BY RANGE (B) (PARTITION 15 <= VALUES < 20, PARTITION OTHERS DYNAMIC THRESHOLD 2),
   (PARTITION VALUES = 30)
   SUBPARTITION BY RANGE (B) (PARTITION VALUES = 20, PARTITION OTHERS),
   (PARTITION OTHERS));
```

Disable dynamic range partitioning on the subpartition on range 10-20 on table A1.

```
ALTER TABLE A1 PARTITION BY RANGE (A)
  ((PARTITION 10 <= VALUES < 20)
   SUBPARTITION BY RANGE (B) (PARTITION 15 <= VALUES < 20, PARTITION OTHERS),
   (PARTITION VALUES = 30)
   SUBPARTITION BY RANGE (B) (PARTITION VALUES = 20, PARTITION OTHERS),
   (PARTITION OTHERS));
```

**Altering annotations**

The following example creates a table with annotations and then alters the annotations:

```
CREATE COLUMN TABLE t1(c1 INT) WITH ANNOTATIONS(
  SET 't_k1'='t_v1', 't_k2'='t_v2'
) COLUMN c1 SET 'c_k1'='c_v1', 'c_k2' = 'c_v2';
ALTER TABLE t1 WITH ANNOTATIONS(
  UNSET ALL SET 'k1' = 'v1', 'k2'='v2'
) COLUMN c1 UNSET ALL SET 'k1'='v1', 'k2'='v2';
```

**NUMA node preferences**

This example immediately unsets the NUMA node preference at the table level. The table is reloaded and allocations are performed by using the HASH scheme available in the attribute engine.

```
ALTER TABLE T1 NUMA NODE NULL IMMEDIATE;
```
This example alters column table T1, setting the NUMA node preference at the partition level on NUMA node index 3. Since DEFERRED is specified, it sets only the NUMA preference for the partition; the actual allocations are made when the table is reloaded.

```
ALTER TABLE T1 PARTITION BY RANGE (colint) (PARTITION '0' <= VALUES < '20' NUMA NODE ('3') DEFERRED);
```

This example immediately alters column table T1, setting the NUMA location preference to partition P0, which already exists. The allocations are made immediately, and the table partition is reloaded to node 4.

```
ALTER TABLE T1 ALTER PARTITION P0 NUMA NODE ('4') IMMEDIATE;
```

This example moves table T1 to NUMA node index 4, immediately allocating the entire table to NUMA node index 4:

```
ALTER TABLE T1 NUMA NODE ('4') IMMEDIATE;
```

**Load unit**

Alter the load unit for a table:

```
ALTER TABLE T PAGE LOADABLE CASCADE;
```

Alter the load unit of any range partition in default storage:

```
ALTER TABLE T PARTITION BY RANGE (C1) (PARTITION 0 <= VALUES < 10 PAGE LOADABLE, PARTITION OTHERS DEFAULT LOADABLE);
```

Add a range partition and specify the load unit:

```
ALTER TABLE T ADD RANGE (C1) (PARTITION 0 <= VALUES < 10, PARTITION OTHERS COLUMN LOADABLE);
```

Alter the load unit for a range partition:

```
ALTER TABLE T ALTER PARTITION 10 DEFAULT LOADABLE;
```

Altering the load unit for a column:

```
ALTER TABLE T ALTER (C1 INT DEFAULT LOADABLE);
```

**Related Information**

- Table Partitioning
- Non-heterogeneous Alter Partition Clauses [page 630]
- Heterogeneous Alter Partition Clauses [page 640]
- CREATE SEQUENCE Statement (Data Definition) [page 810]
- SELECT Statement (Data Manipulation) [page 1104]
- CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 741]
- ALTER CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 435]
4.10.1.22.1 Non-heterogeneous Alter Partition Clauses

Modifies the partitions of an existing table with a non-heterogeneous partitioning schema.

**Syntax**

```sql
ALTER TABLE <table_name>
  [<partition_table_clause>]
  [<add_range_clause>]
  [<redefine_range_partition_clause>]
  [<move_partition_clause>]
  [<drop_range_partition_clause>]
  [<merge_partition_clause>]
  [<alter_dynamic_range_clauses>]
  [<alter_range_partition_attributes_clause>]
  [<alter_partition_numa_node_preference_clause>]
```

**partition_table_clause**

Partitions an existing unpartitioned table using a non-heterogeneous partitioning schema.

```sql
<partition_table_clause> ::=  
  { PARTITION BY <hash_partition> [, , <range_partition> | <hash_partition> ] } 
  { ONLINE ]
  | PARTITION BY <range_partition> [, , <range_partition> ] [ ONLINE ]
  | PARTITION BY <roundrobin_partition> [ , <range_partition> ]
  | [ WITH PARTITIONING ON ANY COLUMNS { ON | OFF } ]
  | [ FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS {ON | OFF} ]
  | [ FOR DEFAULT STORAGE {ALL | NON CURRENT} PARTITIONS {PAGE | COLUMN} ]
  | LOADABLE]
```

Partitioning by date values requires the format YYYYMMDD, YYYY-MM-DD, or YYYY/MM/DD. Date is the most granular date time value that you can use for partitioning.

You can find more information about table partitioning in the *SAP HANA Administration Guide*. 
**hash_partition**

Partitions the created table by using a hash partitioning scheme.

```sql
<hash_partition> ::= HASH ( <partition_expression> [ , <partition_expression> [ , ... ] ] ) PARTITIONS { <num_partitions> | GET_NUM_SERVERS() }
```

**range_partition**

Partitions the created table by using a first-level range partitioning scheme.

```sql
<range_partition> ::= RANGE ( <partition_expression> ) ( <part_range> [ , <part_range> [ , ... ] ] )
```

**subrange_partition**

Partitions the created table by using a range partitioning scheme.

```sql
<subrange_partition> ::= RANGE ( <partition_expression> ) ( <part_subrange> [ , <part_subrange> [ , ... ] ] )
```

**roundrobin_partition**

Partitions the created table by using a round-robin partitioning scheme.

```sql
<roundrobin_partition> ::= ROUNDROBIN PARTITIONS { <num_partitions> | GET_NUM_SERVERS() }
```

**partition_expression**

Declares the specifier that segregates data into partitions.

```sql
<partition_expression> ::= <column_name> [ AS <dynamic_range_data_type> ] | YEAR( <part_column_name> ) | MONTH( <part_column_name> ) | HOUR( <part_column_name> )
```

When using dynamic range partitioning on the OTHER partition, only `<column_name>` is supported.

- **column_name** Specifies the partitioning column. The column must be of a data type supported for the partitioning scheme.

  **Hash partitioning**

  TINYINT, SMALLINT, INT, BIGINT, DECIMAL, DECIMAL(p,s), CLOB, NCLOB, SHORTTEXT, VARCHAR, NVARCHAR, BLOB, VARBINARY, TIME, TIMESTAMP and SECONDDATE. Memory LOBs (ST_MEMORY_LOB type only) are supported but not disk LOBs.

  **Range partitioning**

  STRING, TINYINT, SMALLINT, INT, BIGINT (not supported for dynamic range partitioning), SHORTTEXT, NVARCHAR, DECIMAL(p,s), DATE, TIMESTAMP, SECONDDATE, FIXED, RAW (SQL Binary/Varbinary). Memory LOBs (ST_MEMORY_LOB type) are supported but not disk LOBs.

  **dynamic_range_data_type**
Specifies the data type for dynamic range partitioning.

\[
\text{<dynamic_range_data_type>} ::= \{ \text{INT} \mid \text{DATE} \}
\]

YEAR / MONTH / HOUR

Specifies the precision of the date based partitioning column. When partitioning by HOUR, the \text{<part_range>} must be expressed using one of the following formats:

- \text{YYYY-MM-DD HH}'
  - Enclosed in single quotes.
  - Date delimiter is present.
  - A space exists before the hour granularity.
  - Example: '2010-01-01 23'

- \text{YYYYMMDDHH}'
  - Enclosed in single quotes.
  - No date delimiter is present.
  - No space exists before the hour granularity.
  - Example: '2010010123'

- \text{YYYYMMDDHH}
  - Without single quotes.
  - No date delimiter is present.
  - No space exists before the hour granularity.
  - Example: 2010010123

\text{num_partitions}

Specifies the number of HASH partitions to create for the table.

\[
\text{<num_partitions>} ::= \text{<unsigned_integer>}
\]

\text{GET_NUM_SERVERS()}

Returns the number of servers/partitions according to table placement. You can find more information on table placement in the \text{SAP HANA Administration Guide}.

\text{part_location}

Specifies the location of each HASH partition.

\[
\text{<part_location>} ::= \{ \text{AT LOCATION ( '<HANA\_host>:<HANA\_port>' ) } \}
\]

If not specified, locations are assigned using round robin fashion.

\text{ONLINE}

Allows you to partition an existing table without blocking DML while the DDL is executing. ONLINE is only supported for column-store, non-replicated tables, where specifying ONLINE performs an operation that only acquires a shared table lock. The operation is still serialized with other concurrent DDL operations on the table, including other ONLINE DDL operations. The default value must be a constant value and not a function.

\text{part_range}

Specifies the range specifications for a first-level range partition.
PARTITION <range_values> [ <range_prop_list> ] [, PARTITION <range_values> [ <range_prop_list> ] [,...] ]

range_values
Specifies the values of the partition.

<range_values> ::= <min_value> <= VALUES < <max_value> | VALUE[S] = <target_value> | <partition_others>

Return to <add_range_clause>, <redefine_range_partition_clause>, or <drop_range_partition_clause>

min_value
Specifies the minimum value of the range. The value cannot be negative.

<min_value> ::= <string_literal> | <numeric_literal>

max_value
Specifies the maximum value of the range. The value cannot be negative.

<max_value> ::= <string_literal> | <numeric_literal>

target_value
Specifies a single value of the range. The value cannot be negative.

<target_value> ::= <string_literal> | <numeric_literal>

partition_others
Specifies a partition for all values outside of the defined partition ranges.

<partition_others> ::= OTHERS [ DYNAMIC [ THRESHOLD <threshold_count> ] ]

OTHERS can be defined on a range partition at either level.

DYNAMIC THRESHOLD threshold_count The keyword DYNAMIC enables dynamic range partitioning on the table (default is inactive) and THRESHOLD specifies the maximum row count in the partition before generating a new dynamic range partition from OTHERS or the priority for determining the maximum row count value. The THRESHOLD default, if not specified, is 100 000 000 rows. You can define DYNAMIC on a single RANGE partition or on the second level range of an X-RANGE partition. For more information on dynamic range partitioning, see Dynamic Range Partitioning.

time_selection
Specifies the CURRENT partition for aging.

<time_selection> ::= VALUE = '00000000' IS CURRENT

CURRENT is required for tables with time selection partitioning. Time selection is supported for a single level range partitioned table or a range-range or hash-range partitioned table. Time selection
is not supported for hash-hash or roundrobin-range partitioned tables. The `<data_type>` of the time selection column must be NVARCHAR(8). The `<time_selection>` clause:

- must be defined as the first partition in the range
- can reside on the first level for a single-level range partitioned table or on the second-level for range-range or hash-range partition scheme
- be a single value partition with the value 00000000

`<time_selection>` partitioning also requires the WITH PARTITIONING ON ANY COLUMNS clause set to ON.

Return to `<redefine_range_partition_clause>`.

`range_prop_list`

Specifies the properties for the partition.

```
<range_prop_list> ::= <range_prop> [ <range_prop> ... ] ]
<range_prop> ::= { <persistent_memory_spec_clause> | <numa_node_preference_clause> }
```

Return to `<redefine_range_partition_clause>` or `<numa_node_preference_clause>`.

`persistent_memory_spec_clause`

Enables or disables persistent memory storage preference at the table, range partition, or column level, depending on where the clause is situated in the CREATE statement. For example, when specified inside the `<column_definition>` clause, it enables or disables persistent memory storage for the column.

```
<persistent_memory_spec_clause> ::= PERSISTENT MEMORY <pm_preference>  
<pm_preference> ::= { ON | OFF }
```

`add_range_clause`

Adds a range to an existing partition. Only one range can be added at a time.

```
<add_range_clause> ::= ADD PARTITION ( <column_name> ) USING DEFAULT STORAGE 
( { <range_values> | OTHERS } )
```

`OTHERS`

Specifies a partition for all values outside of the defined partition ranges.

`redefine_range_partition_clause`

Modifies the existing ranges or range properties within a non-heterogeneous partitioned table. You can add new ranges or split or merge existing ranges, as long and the redefined ranges provide a valid partition for all existing data within the table. If a partition is empty, and it is not included in the redefined ranges, then it is dropped.

```
<redefine_range_partition_clause> ::= 
PARTITION BY RANGE (<column_name>) ( <redefine_spec> [ ,<redefine_spec> ] )
```

`redefine_spec`

Specifies the redefined range.

```
<redefine_spec> ::= { [ PARTITION <time_selection> ] | PARTITION 
<range_values> [ , PARTITION <range_values> [,...]]
```
move_partition_clause

Moves an existing non-heterogeneous range partition to a new location.

```plaintext
<move_partition_clause> ::=      MOVE [PARTITION <part_num>[,...] TO <indexserver_host_port> [,...] [PARTITION <part_num>[,...]] TO <indexserver_host_port>[,...]] [PHYSICAL] [ONLINE]
```

The ONLINE parameter can be specified only when moving one or more partitions to different hosts. For example, you could include it when moving partitions 1 and 2 to MyServer1 and partitions 3 and 4 to MyServer2.

```sql
ALTER TABLE T1 MOVE PARTITION 1,2 TO 'MyServer1:34240', PARTITION 3,4 TO 'MyServer2' ONLINE
```

But not when moving just partitions 1 and 2 to MyServer1.

```sql
ALTER TABLE T1 MOVE PARTITION 1,2 TO MyServer1:34240;
```

**part_num**

Specifies the number of the partition to be moved.

```plaintext
<part_num> ::= <unsigned_integer>
```

**indexserver_host_port**

Specifies the internal indexserver host name and port number where the table is to be moved.

```plaintext
@indexserver_host_port> ::= 'host_name:port_number'
```

**ONLINE**

Allows you to move a target table or partition without blocking DML while the DDL is executing.

**PHYSICAL**

The PHYSICAL keyword is only for column store tables. When specified, LOBs on disk are also moved. When not specified, LOBs on disk are excluded from the move.

drop_range_partition_clause

Drops a non-heterogeneous partition from a table. Only one partition can be dropped at a time.

```plaintext
<drop_range_partition_clause> ::= DROP PARTITION { <range_values> | OTHERS }
```

merge_partition_clause

Merges all parts of a partitioned table into a non-partitioned table.

```plaintext
<merge_partition_clause> ::= MERGE PARTITIONS
```

alter_dynamic_range_clauses

Defines the alter operations you can perform to dynamic range options.

Dynamic range partitioning is only supported for single RANGE and X-RANGE partitions. For X-RANGE, dynamic range can only be defined on the second-level partition. Dynamic range partitioning is disabled by default. PARTITION OTHERS must exist on the applicable level before you can execute alter operations on
dynamic range options. For information on dynamic range partitioning, see Dynamic Range Partitioning in the SAP HANA Administration Guide.

**add_dynamic_range**

Creates a new partition.

```sql
<add_dynamic_range> ::= ADD PARTITION FROM OTHERS
```

**drop_dynamic_range**

Drops empty partitions.

```sql
<drop_dynamic_range> ::= DROP EMPTY PARTITIONS
```

**partition_others_dynamic_range**

Specifies other supported dynamic partitioning options.

```sql
<partition_others_dynamic_range> ::= PARTITION OTHERS { DYNAMIC
[ THRESHOLD <threshold_count> ] | NO DYNAMIC }
```

The keyword DYNAMIC enables dynamic range partitioning on the table (default is inactive) and THRESHOLD specifies the maximum row count in the partition before generating a new dynamic range partition from OTHERS or the priority for determining the maximum row count value. The THRESHOLD default, if not specified, is 100 000 000 rows. You can define DYNAMIC on a single RANGE partition or on the second level range of an X-RANGE partition.

**alter_range_partition_attributes_clause**

Modifies the attributes on the partition.

```sql
<alter_range_partition_attributes_clause> ::=    { [ FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS { ON | OFF } ]
| [ WITH PARTITIONING ON ANY COLUMNS { ON | OFF } ]
| [ FOR DEFAULT STORAGE {ALL | NON CURRENT} PARTITIONS { PAGE | COLUMN } LOADABLE ]

WITH PARTITIONING ON ANY COLUMNS
Required for time selection. Disabled (OFF) by default.

FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS
This clause is only supported for time selection and it specifies where the uniqueness check for rows entered or altered is performed. If OFF (default), then only the current (hot) partition is checked. If ON, then both current and non-current (cold) partitions are checked.

FOR DEFAULT STORAGE {ALL | NON CURRENT} PARTITIONS {PAGE | COLUMN} LOADABLE
This clause is only supported for time selection and specifies the load method for current and non-current partitions.

**alter_partition_numa_node_preference_clause**

Alters the NUMA node preference for the specified partition number.

```sql
<alter_partition_numa_node_preference_clause> ::= ALTER PARTITION <part_num>
<numa_node_preference_clause>
```

**part_num**
The number of the partition you are changing the NUMA node preferences for. To determine the value, use the TABLE_PARTITIONS system view.

```sql
numa_nodePreference_clause
```

Alters the NUMA node preferences. Although this clause is defined here at the table level, `<numa_node_preference_clause>` can be set in various locations such as range partition definitions (not hash or round-robin) and column definitions, as indicated in the syntax within the topic. `<numa_node_preference_clause>` is not supported for heterogeneous partitions.

```sql
<numa_node_preference_clause> ::= NUMA NODE { ( <numa_node_index_spec> ) | NULL } [ IMMEDIATE | DEFERRED ]
<numa_node_index_spec> ::= <numa_node_spec> [, <numa_node_spec> [ ,... ] ]
<single_node_spec> ::= <integer_const>
<range_node_spec> ::= <integer_const> TO <integer_const>
```

NULL

To unset existing NUMA node preferences on the associated object (table, column, partition, or replica), specify NUMA NODE NULL.

IMMEDIATE, DEFERRED

Specify whether to apply the NUMA node preferences immediately (IMMEDIATE) or to defer applying the preferences until the next time the table is reloaded (DEFERRED). If you specify IMMEDIATE, then the associated table, column, or table partition is immediately moved according to the altered preference settings.

The IMMEDIATE option is not effective if the `<alter_persistent_memory_spec_clause>` (Fast Restart or PMEM) is enabled on this table; in this case the new NUMA node will only be used after the column is rewritten into persistent memory. You can force this by either running a delta merge `<auto_merge_option>` or explicitly deleting persistent memory so that it is rewritten with the next load: UNLOAD `<table>` DELETE PERSISTENT MEMORY;

```sql
integer_const
```

<integer_const> cannot be a negative number.

numa_node_spec

Specify one or more single NUMA nodes (<single_node_spec>), or one or more NUMA node ranges (<range_node_spec>), or a mixture of both.

NUMA node indexes should be specified in the range of 0 to one less than max_numa_node_count, where max_numa_node_count is the number of NUMA nodes configured for the system. If the NUMA node index specified is greater than or equal to max_numa_node_count, then a random NUMA node in the range of 0 to one less than max_numa_node_count is selected for allocation. For example, on a system where max_numa_node_count is set to 8, if you specify a NUMA node index of 10, then any node in the range of 0 to 7 (inclusive) is chosen randomly for allocation.

Return to `<range_prop_list>`.

Description

Modifies the partitions of an existing table with a non-heterogeneous partitioning schema. To modify partitions of an existing table with a heterogeneous partitioning schema, see the topic Heterogeneous Partition Clauses in this guide.
Permissions

The (non-destructive) clauses listed below can be executed with the PARTITION ADMIN system privilege. For all other clauses, you must have the ALTER object privilege on the table to alter a partition.

- `<partition_table_clause>`
- `<add_range_clause>`
- `<redefine_range_partition_clause>`
- `<move_partition_clause>`
- `<merge_partition_clause>`
- `<alter_dynamic_range_clauses>`
- `<alter_range_partition_attributes_clause>`
- `<alter_partition_numa_node_preference_clause>`

Examples

Convert a non-partitioned column store table named T1 to a range partitioned table.

```sql
CREATE COLUMN TABLE T1 (a INT, b INT);
ALTER TABLE T1 PARTITION BY RANGE (a) ( PARTITION 10 <= VALUES < 20);
```

Add a new second-level partition to an existing range-partitioned table, making it a range-range partitioned table.

```sql
ALTER TABLE T1 PARTITION BY
  RANGE (a) ( PARTITION 10 <= VALUES < 20),
  RANGE (b) ( PARTITION 100 <= VALUES < 150);
```

Redefine the existing first-level partition to ranges 10 - 15 and 15 - 20.

```sql
ALTER TABLE T1 PARTITION BY
  RANGE (a) ( PARTITION 10 <= VALUES < 15, PARTITION 15 <= VALUES < 20),
  RANGE (b) ( PARTITION 100 <= VALUES < 150);
```

Add a new partition to the second-level partition.

```sql
ALTER TABLE T1 ADD PARTITION (b) USING DEFAULT STORAGE (150 <= VALUES < 200);
```

Move a partition.

```sql
ALTER TABLE T1 MOVE PARTITION 1 TO 'myhost:30201';
```

Move multiple partitions to new locations.

```sql
ALTER TABLE T2 MOVE PARTITION 1,2 TO 'myhost2:34240', PARTITION 3,4 TO 'myhost3:34242' ONLINE;
```
Drop second-level partition range 10 - 15.

```
ALTER TABLE T1 DROP PARTITION (a) 10 <= VALUES < 15;
```

Merge all partitions into a single non-partitioned table.

```
ALTER TABLE T1 MERGE PARTITIONS;
```

Alter column table T1, setting the NUMA node preference at the partition level on NUMA node index 3. Since DEFERRED is specified, it sets only the NUMA preference for the partition; the actual allocations are made when the table is reloaded. IMMEDIATE alters column table T1, setting the NUMA node preference on partition 1, which already exists. The allocations are made immediately, and the table partition is reloaded to node 4.

```
ALTER TABLE T1 ALTER PARTITION 1 NUMA NODE ('4') IMMEDIATE;
```

Create a non-partitioned table named T2, convert the table to a range partitioned table, and then add an additional partition.

```
CREATE COLUMN TABLE T2 (a INT, b INT);
ALTER TABLE T2 PARTITION BY RANGE (a) (PARTITION VALUE = 1, PARTITION OTHERS);
ALTER TABLE T2 ADD PARTITION (a) 2 <= VALUES < 10;
```

Create a non-partitioned table named T3 and then convert the table to a range-range partitioned table.

```
CREATE COLUMN TABLE T3 (a INT, b INT);
ALTER TABLE T3 PARTITION BY RANGE (a) (PARTITION VALUE = 1, PARTITION OTHERS), RANGE (b) (PARTITION 2 <= VALUES < 10);
```

Convert an existing hash partitioned table named T4 to a range partitioned table.

```
CREATE TABLE T4 (A INT, B INT) PARTITION BY HASH (A) PARTITIONS 2;
ALTER TABLE T4 PARTITION BY RANGE (A) (PARTITION 10 <= values < 20, PARTITION OTHERS);
```

Convert an existing range partitioned table named T5 to a hash partitioned table.

```
CREATE TABLE T5 (A INT, B INT) PARTITION BY RANGE (A) (PARTITION 10 <= values < 20, PARTITION OTHERS);
ALTER TABLE T5 PARTITION BY HASH (A) PARTITIONS 2;
```

Convert a range-range partitioned (time selection) table named T6 to a range partitioned table (convert RANGE-RANGE to RANGE).

```
CREATE COLUMN TABLE T6 (A INT PRIMARY KEY, B INT, _DATAAGING NVARCHAR(8))
PARTITION BY RANGE(A) (PARTITION VALUE = 1, PARTITION 10 <= VALUES < 20, PARTITION OTHERS), RANGE (_DATAAGING)
(USING DEFAULT STORAGE (PARTITION value = '00000000' IS CURRENT, PARTITION '20100101' <= VALUES < '20110101', PARTITION OTHERS))
WITH PARTITIONING ON ANY COLUMNS ON FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS OFF FOR DEFAULT STORAGE NON CURRENT PARTITIONS PAGE LOADABLE;
ALTER TABLE T6 PARTITION BY RANGE(A) (PARTITION VALUE = 1, PARTITION 10 <= VALUES < 20, PARTITION OTHERS);
```
**Dynamic Range Partitioning**

These examples are based on table A1 using this CREATE TABLE statement.

```
CREATE COLUMN TABLE A1 (A INT, B INT NOT NULL) PARTITION BY RANGE (A)
  (PARTITION VALUES = 10),
  RANGE (B) (PARTITION VALUES = 20, PARTITION OTHERS);
```

Enable dynamic range partitioning on the OTHERS partition.

```
ALTER TABLE A1 PARTITION others dynamic;
```

Add another partition from OTHERS.

```
ALTER TABLE A1 ADD PARTITION FROM OTHERS;
```

Drop any empty OTHERS partitions.

```
ALTER TABLE A1 DROP EMPTY PARTITIONS;
```

Change the dynamic threshold to 2, which uses the value defined in the DYNAMIC_RANGE_THRESHOLD field of the TABLE_PLACEMENT table.

```
ALTER TABLE A1 PARTITION OTHERS DYNAMIC THRESHOLD 2;
```

Disable dynamic range partitioning on the OTHERS partition.

```
ALTER TABLE A1 PARTITION OTHERS NO DYNAMIC;
```

**Related Information**

- Heterogeneous Alter Partition Clauses [page 640]
- ALTER TABLE Statement (Data Definition) [page 589]
- Table Partitioning
- TABLE_PARTITIONS System View [page 1681]
- Dynamic Range Partitioning

**4.10.1.22.2 Heterogeneous Alter Partition Clauses**

Modifies the partitions of an existing table with a heterogeneous partitioning schema.

**Syntax**

```
ALTER TABLE <table_name>
  [<partition_table_clause>]
```
partition_table_clause

Partitions an existing unpartitioned table using a heterogeneous range, range-range, or range-hash partitioning scheme.

```
<partition_table_clause> ::= PARTITION BY RANGE (<partition_expression>)      
                          [ [ NO ] PRIMARY KEY CHECK ] ( ... ]      
                          | ( <part_range> ) 
                          <subpartition_by_clause> ::= SUBPARTITION BY ( sub_part_range | 
                          [ <sub_part_hash> ]
```

All subpartitions must be of the same type and reference the same column or in the case of a hash subpartition the same group of columns. Mixed range and hash subpartitions are not supported. If the first subpartition is a range using column B, then all additional subpartitions must be ranges using column B. If the first subpartition is a hash, referencing columns B and C, then all subpartitions must hash referencing the same group of columns.

partition_expression

Declares the specifier that segregates data into partitions.

```
<partition_expression> ::=<column_name>
                         | YEAR( <column_name> )      
                         | MONTH( <column_name> )      
                         | HOUR( <column_name> )
```

To use dynamic range partitioning, only `<column_name>` is supported for the partitioning column containing the OTHERS partition.

`column_name` Specifies the partitioning column. The column must be of a data type supported for the partitioning scheme.

Hash partitions

TINYINT, SMALLINT, INT, BIGINT, DECIMAL, DECIMAL(p,s), CLOB, NCLOB, SHORTTEXT, VARCHAR, NVARCHAR, BLOB, VARBINARY, DATE, TIME, TIMESTAMP and SECONDDATE.

Range partitions

STRING, TINYINT, SMALLINT, INT, BIGINT (not supported for dynamic range partitioning), SHORTTEXT, NVARCHAR, DECIMAL(p,s), DATE, TIMExTIME, SECONDDATE, FIXED, RAW (SQL Binary/Varbinary). Memory LOBs (ST_MEMORY_LOB type) are supported but not disk LOBs.

YEAR / MONTH Specifies the precision of the date based partitioning column.

[ NO ] PRIMARY KEY CHECK
The primary key check is performed on the first level of a partition. If not specified, the default behavior is to not perform the check. PRIMARY KEY CHECK is not supported on multi-store tables or in combination with the time selection feature.

**part_range**

Specifies the range for the first- or second-level range partition.

```
<part_range> ::= PARTITION <range_values> [ INSERT {OFF | ON} ]
```

Return to `<add_partition_clauses>`.

**range_values**

Specifies the values of the first- or second-level range partition.

```
<range_values> ::=      <min_value> <= VALUES < <max_value>     | VALUE[S] = <target_value>     | <partition_others>
```

Return to `<redefine_partition_clause>` , `<move_partition_clauses>` , `<drop_partition_clauses>` , `<alter_partition_load_unit_clause>` , or `<alter_partition_group_clauses>`.

**min_value**

Specifies the minimum value of a first- or second-level range partition. The value cannot be negative.

```
<min_value> ::= <string_literal> | <numeric_literal>
```

**max_value**

Specifies the maximum value of a first- or second-level range partition. The value cannot be negative.

```
<max_value> ::= <string_literal> | <numeric_literal>
```

**target_value**

Specifies a single value of a first- or second-level range partition. The value cannot be negative.

```
<target_value> ::= <string_literal> | <numeric_literal>
```

**partition_others**

Specifies a partition for all values outside of the defined partition ranges.

```
<partition_others> ::= OTHERS [ DYNAMIC [ THRESHOLD <threshold_count> ] ] | [ DYNAMIC INTERVAL <interval> ]
```

**DYNAMIC THRESHOLD threshold_count** The keyword DYNAMIC enables dynamic range partitioning on the table (default is inactive) and THRESHOLD specifies the maximum row count in the partition before generating a new dynamic range partition from OTHERS or the priority for determining the maximum row count value. The THRESHOLD default, if not specified, is 100 000 000 rows. You can only define DYNAMIC on a single range partition or on any range subpartition. For more information on dynamic range partitioning, see *Dynamic Range Partitioning*.  

642  PUBLIC

SAP HANA SQL Reference Guide for SAP HANA Platform

SQL Reference
DYNAMIC INTERVAL

Specifies a dynamic interval for the range OTHERS partition. The interval is between the last range partition and new partition created dynamically.

```
interval ::= <interval_value> <interval_type>
<interval_value> ::= <unsigned_integer>
<interval_type> ::= {YEAR | MONTH | HOUR}
```

Dynamic interval is only supported when the partition column type is TINYINT, SMALLINT, INT, BIGINT, DATE, SECONDDATE or LONGDATE. If no `<interval_type>` is specified, INT is used implicitly.

**INSERT { ON | OFF }

Specifies if INSERT statements are allowed on a partition. If defined at the first-level, it applies to all second-level partitions within the first-level partition. If defined at both the first- and second-level, any value at the second-level overrides the first-level value. If not specified, default is ON.

**sub_part_range

Specifies the range subpartition.

```
<sub_part_range> ::= RANGE ( <sub_range_col_def> ) ( <part_range> [,...] )
<sub_range_col_def> ::= { <exist_subpart_col_name> | <partition_expression> }
<part_range> ::= 
```

All subpartitions must be of the same type and reference the same column or in the case of a hash subpartition the same group of columns. Mixed range and hash subpartitions are not supported. If the first subpartition is a range using column B, then all additional subpartition must be ranges using column B. If the first subpartition is a hash, referencing columns B and C, then all subpartitions must hash referencing the same group of columns.

**exist_part_column_name** Specifies the name of the existing first-level range partitioning column.

**sub_range_col_def**

Specifies the subpartitioning column. If this is the first subpartition in the partitioned table, then specify `<partition_expression>`. For all subsequent range subpartitions added or referenced, specify `<exist_subpart_col_name>`, the name of the existing range subpartitioning column.

Return to `<partition_table_clause>`, `<add_partition_clauses>` or `<redefine_partition_clause>`.

**sub_part_hash

Specifies the hash subpartition.

```
<sub_part_hash> ::= HASH ( <sub_hash_col_def> ) PARTITIONS <num_partitions> [ AT [ LOCATION ] ( <loc_list> ) ]
<sub_hash_col_def> ::= { <exist_col_grp_list> | <partition_expression> [,<partition_expression> [,...] ] }
```

All subpartitions must be of the same type and reference the same column or in the case of a hash subpartition the same group of columns. Mixed range and hash subpartitions are not supported. If the first subpartition is a range using column B, then all additional subpartition must be ranges using column B. If the first subpartition is a hash, referencing columns B and C, then all subpartitions must hash referencing the same group of columns. With the exception of LOCATION, all hash partitions share the same properties.
sub_hash_col_def

Specifies the subpartitioning column. If this is the first subpartition in the partitioned table, then specify `<partition_expression>`. For all subsequent hash subpartitions added or referenced, specify `<exist_col_grp_list>`, the name of the existing hash partitioning column or group of columns.

loc_list

Specifies where the hash partition resides.

**Note**

When the number of target partitions is a multiple of the number of source partitions, they are distributed across the available index servers in a round-robin style. However, in cases where the number of target partitions is not a multiple of the source partitions, all the target partitions are stored at the location of the first source partition.

```
<loc_list> ::= <location> [ , <location> [,...] ]
<location> ::= '<HANA_host>:<HANA_port>'
```

Return to `<partition_table_clause>`, `<add_partition_clauses>` or `<redefine_partition_clause>`.

add_partition_clauses

Defines the alter operations to add first- and second-level partitions to an existing partitioned table.

add_partition_clause

Adds a new first-level partition, with an optional subpartition, to an existing partitioned table.

```
<add_partition_clause> ::= ADD PARTITION RANGE ( <exist_part_column_name> ) ( ( <part_range_list> ) [ SUBPARTITION BY { <sub_part_range> | <sub_part_hash> } ] );
<part_range_list> ::= <part_range_list> [ , <part_range> [,...] ]
<exist_part_column_name> is the name of the existing first-level range partitioning column.
```

You can also add a partition by redefining existing partition ranges. See `<redefine_partition_clause>`.

add_subpart_clause

Adds a new range subpartition to an existing first-level range partition.

```
<add_partition_clause> ::= ALTER PARTITION RANGE ( <exist_part_column_name> ) ( ( <part_range_list> ) )
PARTITION BY RANGE ( <sub_range_column_def> ) ( ( <subpart_range_list> ) ) ;
<part_range_list> ::= <part_range> [ , <part_range> [,...] ]
<sub_range_col_def> ::= { <exist_subpart_col_name> | <partition_expression> }
<subpart_range_list> ::= <part_range> [ , <part_range> [,...] ]
```

If data exists in the table, then the subpartition specifications must account for all data within the partitioning column. For example, if a table contains the data rows (10, 100) and (10, 200), and the first-level partitioning column is the first column, then the subpartition range must include both values of column two. You could specify a range (100 - 200) or fixed values (100, 200). You could also include the OTHERS partition to account for later values added, which might fall outside the subpartition range.
To add a hash subpartition to an existing first-level range partition, use the `<redefine_partition_clause>` syntax.

**sub_range_col_def**
Specifies the subpartitioning column. If this is the first subpartition in the partitioned table, then specify `<partition_expression>`. For all subsequent range subpartitions added or referenced, specify `<exist_subpart_col_name>`, the name of the existing range subpartitioning column.

**redefine_partition_clause**
Modifies the existing ranges and the INSERT property of the first- and second-level partitions. The new ranges must provide a valid partition for all existing data within the table. If a partition or subpartition is empty, and it is not included in the redefined ranges, then it is dropped.

```
<redefine_partition_clause> ::= PARTITION BY RANGE
  ( <exist_partition_column_name> )
  ( ( <redefine_part_range_list> ) [ SUBPARTITION BY { <sub_part_range> | <sub_part_hash> } )
<redefine_part_range_list> ::= <redefine_part_range> [, <redefine_part_range> [,...:]]
```

**exist_partition_column_name** Specifies the name of the existing first-level range partitioning column.

**redefine_part_range**
Specifies the range values for the redefined first-level range partition.

```
<redefine_part_range> ::= PARTITION <range_values> [ INSERT {OFF | ON} ]
```

**move_partition_clauses**
Defines the alter operations to move partitions to a new location.

**move_range_part**
Move a first- or second-level range partition to a new location.

```
<move_range_part> ::= MOVE PARTITION <partition_number>[,.<partition_number>][,...:]
  TO ( <host>:<port> )
  [ PARTITION <partition_number>[,.<partition_number>][,...:]
  TO ( <host>:<port> ) [ ONLINE ] ]
  | MOVE PARTITION <move_range_clause> TO ( <host>:<port> )
```

The ONLINE parameter can be specified only when moving one or more partitions to different hosts. For example, you could include it when moving partitions 1 and 2 to MyServer1 and partitions 3 and 4 to MyServer2.

```
ALTER TABLE T1 MOVE PARTITION 1,2 TO 'MyServer1:34240', PARTITION 3,4 TO 'MyServer2' ONLINE
```

But not when moving just partitions 1 and 2 to MyServer1.

```
ALTER TABLE T1 MOVE PARTITION 1,2 TO MyServer1:34240;
```

If you specify only the parent first-level partition, then all child partitions are also moved to the same location. If you specify a subpartition, then only the subpartition is moved.

**partition_number**
Specifies the first-level partition number to move. You cannot move only a second-level partition by partition number. To determine the value, use the TABLE_PARTITIONS system view.

**move_range_clause**

Specifies the first-or second-level range partition to move.

```
<move_range_clause> ::= RANGE ( <exist_partition_column_name> ) ( <move_range_spec> [, <move_range_spec> [,...] ] )
<move_range_spec> ::= ( <move_range_list> ) [ SUBPARTITION BY RANGE ( <exist_subpart_column_name> ) ( <move_range_list> ) ]
<move_range_list> ::= <move_range> [, <move_range> [,...] ]
<move_range> ::= PARTITION <range_values>
```

**exist_partition_column_name** Specifies the name of the existing first-level range partitioning column.

**exist_subpart_column_name** Specifies the name of the existing range subpartitioning column.

**move_hash_part**

Move a hash subpartition to a new location.

```
<move_range> ::= MOVE SUBPARTITIONS OF <move_hash> TO ( <move_loc_list> );
```

**move_hash**

Specifies the range partitions containing the hash partitions to move.

```
<move_hash> ::= RANGE ( <exist_partition_column_name> ) ( ( PARTITION <range_values> ) )
```

**move_loc_list**

Specifies the new locations for the hash subpartition.

```
<move_loc_list> ::= '<move_loc>' [, '<move_loc>' [,...] ]
<move_loc>::= <host>:<port>
```

Locations are assigned in the order they are specified. If you specify more locations than available hash partitions, unused locations are ignored. If you specify less locations than available hash partitions, the location restarts with the first location in the list. For example, if there are three hash partitions but only two locations are specified, partition three is assigned to location one.

For a hash subpartition with multiple partitions, you cannot move a single partition. The location list applies to all partitions, assigned in the order listed, but moving a partition to its same location results in the partition remaining stationary. To move specific partitions to specific locations, order the list a locations to match the order of the partitions. For example, the hash subpartition has three partitions, assigned to locA, locB, and locC respectively. To move partition 1 only, list the locations as locD, locB, locC. Partition 1 moves from locA to locD and partitions 2 and 3 move to their same location, effectively remaining unmoved.

**drop_partition_clauses**

Defines the alter operations to drop range partitions. Hash partitions cannot be dropped.

**drop_partition_clause**
Drops a first-level range partition and all associated subpartitions. Dropping a partition deletes any data contained within the partition or associated subpartition.

```sql
<drop_part_range> ::= DROP PARTITION RANGE (<exist_part_column_name>) ( ( <drop_part_range> [, <drop_part_range> ] ) )
```

**exist_part_column_name** Specifies the name of the existing first-level range partitioning column.

**drop_part_range** Specifies the first-level range being dropped.

```sql
<drop_part_range> ::= PARTITION <range_values>
```

**drop_subpart_clause**
Drops a range subpartition. Dropping the last range subpartition deletes the first-level range partition as well.

```sql
<drop_subpart_clause> ::= ALTER PARTITION RANGE (<exist_part_column_name>) ( ( <part_range_drop> ) ) DROP PARTITION RANGE (<exist_part_column_name>) ( ( <range_values> [, <range_values> [,...] ] ) )
```

**exist_part_column_name** Specifies the name of the existing first-level range partitioning column.

**part_range_drop** Specifies the first-level range partition containing the range subpartition being dropped.

```sql
<part_range_drop> ::= <drop_part_range>
```

**merge_partition_clause**
Merges all parts of a partitioned table into a non-partitioned table.

```sql
<merge_partition_clause> ::= MERGE PARTITIONS
```

**alter_dynamic_range_clauses**
Defines the alter operations you can perform to dynamic range options.

Dynamic range is only supported for single range and range-range partitions. For range-range partitions, dynamic range partitions can only be defined on subpartitions. Dynamic range partitioning is disable by default. The OTHERS partition must exist on the applicable level before you can alter dynamic range partition options. For information on dynamic range partitioning, see *Dynamic Range Partitioning* in the SAP HANA Administration Guide.

To enable or disable dynamic range partitioning or modify the threshold on an existing subpartition, redefine the partitions and add or remove the keywords DYNAMIC or DYNAMIC THRESHOLD to/from the existing OTHERS partition.

**add_dynamic_range**
Creates a new dynamic range OTHERS partition.

```sql
<add_dynamic_range> ::= ADD PARTITION FROM OTHERS;
```

**drop_dynamic_range**
Drops empty partitions.

\[
\text{\texttt{<drop\_dynamic\_range> ::= DROP EMPTY PARTITIONS;}}
\]

\textbf{alter\_partition\_load\_unit\_clause}

Alters the paging attributes for a range partition or subpartition.

\[
\text{\texttt{<alter\_partition\_load\_unit\_clause> ::= ALTER PARTITION \{<partition\_number> | <range> \} <load\_unit>;}}
\]

\textbf{partition\_number} Specifies the numerical value of a single partition or subpartition. To determine the value, use the TABLE\_PARTITIONS system view.

\textbf{range}

Specifies the partition or subpartition range to alter.

\[
\text{\texttt{<range> ::= RANGE ( <exist\_part\_column\_name> ) ( PARTITION <range\_values> [, PARTITION <range\_values> [...]] )}}
\]

\textbf{exist\_part\_column\_name} Specifies the name of the existing first-level range partitioning column.

\textbf{sub\_range}

Specifies the range of the range subpartition to alter.

\[
\text{\texttt{<sub\_range> ::= RANGE ( <exist\_part\_column\_name> ) PARTITION <range\_values> [, [...]]}}
\]

\textbf{sub\_hash}

Specifies the hash subpartition.

\[
\text{\texttt{<sub\_hash> ::= HASH ( <exist\_part\_column\_name> ) PARTITIONS <num\_partitions>}}
\]

\textbf{load\_unit}

Specifies how to load data into memory when the table is queried. Specifying the load unit is only supported on column-store tables. \texttt{<load\_unit>} can be set at column, table, and partition levels.

\[
\text{\texttt{<load\_unit> ::= \{ COLUMN | PAGE | DEFAULT \} LOADABLE [ CASCADE ]}}
\]

\textbf{COLUMN LOADABLE}

In-memory loading - the entire column is loaded into memory. COLUMN LOADABLE boosts performance at the cost of higher memory usage. This is the default behavior unless another value is inherited.

\textbf{PAGE LOADABLE}

In-buffer cache loading - column data is loaded by page into the buffer cache. PAGE LOADABLE reduces memory usage for specific columns by not requiring those columns to be fully memory resident.

\textbf{DEFAULT LOADABLE}
If you specify DEFAULT, then there is no explicit preference for the paging attribute that is used when the results are loaded from the table. If there is an inherited loading unit from another object, then that loading unit is used.

CASCADE

Specify CASCADE when you want the paging attribute to apply to all objects below the specified object in the object hierarchy, so table paging attributes apply to partitions.

**alter_partition_group_clauses**

Defines the alter operations you can perform with regards to setting and unsetting range partition group options.

```plsql
<alter_partition_group_clauses> ::=      <set_partition_group_partitionid_clause>     | ...     | <set_partion_replica_group_partitionid_clause>     | <unset_partion_replica_group_partitionid_clause>
```

**set_partition_group_partitionid_clause**

Specifies the group options to set on the specified range partition or subpartition.

```plsql
<set_partition_group_partitionid_clause> ::=     ALTER PARTITION <partition_id_list>  SET <group_list>
```

GROUP options are only supported on range partitions in a range, range-range, or range-hash partition schema. Use the M_TABLE_PARTITIONS view to see effective group values (after applying inheritance from parent levels).

**partition_id_list** Specifies the numerical value of the partition.

```plsql
<partition_id_list> ::= <partition_id> | '(' <partition_id> [,<partition_id>...] ')' 
```

To determine the value, use the TABLE_PARTITIONS system view.

**group_list** Specifies the GROUP option to apply to the specified partition.

```plsql
<group_list> ::= <group> [ <group> ... ]
<group> ::= GROUP {NAME | TYPE | SUBTYPE} <identifier>
```

Each partition can be assigned a group name, type, and subtype. If a GROUP option already exists on a partition, then the new value overwrites the existing value.

**unset_partition_group_partitionid_clause**

Removes all GROUP options (NAME, TYPE, SUBTYPE) applied to the specified partitions.

```plsql
<unset_partition_group_partitionid_clause> ::=     ALTER PARTITION <partition_id_list>  UNSET GROUP
```

**set_partition_group_range_clause**

Specifies the group options to set on the specified range partition or subpartition.

```plsql
<set_partition_group_range_clause> ::= 
```
ALTER PARTITION <group_range> SET <group_list> [ CASCADE ]

GROUP options are only supported on range partitions in a range, range-range, or range-hash partition schema. Use the M_TABLE_PARTITIONS view to see effective group values (after applying inheritance from parent levels).

**group_range**

Specifies the ranges of the range partition to apply the GROUP option to.

```sql
<group_range> ::= RANGE ( <exist_part_column_name> ) ( ( [SUBPARTITION BY RANGE ( <exist_part_column_name> ) ( <subpart_range_list> ) [,...] ) ) [PARTITION <range_values> ] [,...] )
<subpart_range_list> ::= PARTITION <range_values> ] [,...] )
```

**exist_part_column_name** Specifies the name of the existing first-level range partitioning column.

**CASCADE** Applies the property change to all subpartitions within the specified range. This clause is only supported with the `<group_range>` option.

**unset_partition_group_range_clause**

Removes all GROUP options (NAME, TYPE, SUBTYPE) applied to the specified partition.

```sql
<unset_partition_group_range_clause> ::= ALTER PARTITION <group_range> } UNSET GROUP [ CASCADE ]
```

**set_partition_replica_group_partitionid_clause**

Specifies the group options to set on the specified replica partition.

```sql
<set_partition_replica_group_partitionid_clause> ::= ALTER REPLICA PARTITION <partition_id_list> SET <group_list> [AT <replica_locations>] )
```

GROUP options are only supported on range partitions in a range, range-range, or range-hash partition schema. Use the M_TABLE_PARTITIONS view to see effective group values (after applying inheritance from parent levels).

**replica_locations**

Adds the replica to the specified locations.

```sql
<replica_locations> ::= { <indexserver_host_port> | ( <indexserver_host_port>, ... ) )
<indexserver_host_port> ::= '<host_name>:<port_number>'
```

If `<replica_locations>` is not specified, then the replica table is added based on the table placement rule. If no table placement rule is also defined, then the replica table is added one of the slave nodes that is not a source table location and where there is not already an existing replica of the table. If there are no more replica nodes to add a replica table, then an error is returned.

**unset_partition_replica_group_partitionid_clause**

Removes all GROUP options (NAME, TYPE, SUBTYPE) applied to the specified partition id.

```sql
<unset_partition_replica_group_partitionid_clause> ::=
```
ALTER REPLICA PARTITION <partition_id_list> UNSET GROUP [ AT <replica_locations> ]

Description

Modifies the partitions of an existing table with a heterogeneous partitioning schema. To modify partitions of an existing table with a non-heterogeneous partitioning schema, see the topic Non-heterogeneous Partition Clauses in this guide.

Permissions

The (non-destructive) clauses listed below can be executed with the PARTITION ADMIN system privilege. For all other clauses, you must have the ALTER object privilege on the table to alter a partition.

- <partition_table_clause>
- <add_partition_clauses>
- <redefine_partition_clause>
- <move_partition_clauses>
- <merge_partition_clause>
- <alter_dynamic_range_clauses>
- <alter_partition_load_unit_clause>
- <alter_partition_group_clauses>

Examples

Convert the non-partitioned table T1 to a range-range partitioned table. The table has four first-level partitions (10-20,20-30,40,OTHERS). Range 10-20 has three subpartition ranges (0-10, 10-100, others). Range 40 has two subpartitions ranges (0-10, OTHERS). Both subpartitions have dynamic range partitioning enabled.

CREATE COLUMN TABLE T1 (a INT, b INT NOT NULL);
ALTER TABLE T1 PARTITION BY RANGE (a)
  ((PARTITION 10 <= VALUES < 20 INSERT OFF, PARTITION 20 <= VALUES < 30 INSERT OFF)
   SUBPARTITION BY RANGE (b) (PARTITION 0 <= VALUES < 10, PARTITION 10 <= VALUES < 100, PARTITION OTHERS DYNAMIC THRESHOLD 2),
   (PARTITION VALUES = 40)
   SUBPARTITION BY RANGE(B) (PARTITION 0 <= VALUES < 10, PARTITION OTHERS DYNAMIC THRESHOLD 2),
   (PARTITION OTHERS));
Convert the non-partitioned table T2 to a range-hash partitioned table. The first level has two partition ranges (10-20, 20-30). The INSERT OFF property means you can’t add data to the partitions. The second level is a hash with two partitions.

```
CREATE COLUMN TABLE T2 (a INT, b INT);
ALTER TABLE T2 PARTITION BY RANGE (a) ((PARTITION 10 <= VALUES < 20 INSERT OFF, PARTITION 20 <= VALUES < 30 INSERT OFF)
    SUBPARTITION BY HASH (b) PARTITIONS 2);
```

Add a new first-level range partition 41-50 to table T1. The new partition has a subpartition with one range 0-10.

```
ALTER TABLE T1 ADD PARTITION RANGE (a) ((PARTITION 41 <= VALUES < 50 )
    SUBPARTITION by RANGE (b) (PARTITION 0 <= VALUES < 10 ));
```

Redefine the existing first-level partition range 10-20 to ranges 5-10 and 15-20. Each redefined range retains its original subpartition. All existing range values must be accounted for in the redefined structure or they will be dropped if they do not contain data.

```
ALTER TABLE T1 PARTITION BY RANGE (a)
    ((PARTITION 5 <= VALUES < 10, PARTITION 10 <= VALUES < 20 INSERT OFF,
    PARTITION 20 <= VALUES < 30 INSERT OFF)
    SUBPARTITION BY RANGE (b) (PARTITION 0 <= VALUES < 10, PARTITION 10 <= VALUES < 100, PARTITION OTHERS DYNAMIC THRESHOLD 2),
    (PARTITION VALUES = 40) SUBPARTITION BY RANGE (b) (PARTITION 0 <= VALUES < 10),
    (PARTITION OTHERS));
```

Move partition 1 (range 5-10) on table T1 to host myhost on port 30203.

```
ALTER TABLE T1 MOVE PARTITION 1 TO 'myhost:30203';
```

Move range 5-10 with subpartition range 0-10 on table T1 to host myhost on port 30203.

```
ALTER TABLE T1 MOVE RANGE (a) ((PARTITION 5 <= VALUES < 10) SUBPARTITION BY
    RANGE (b) (PARTITION 0 <= VALUES < 10)) TO 'myhost:30203';
```

Move the hash subpartition of range 10-20 on table T2 to host myhost1 port 30203 and myhost2 port 30203.

```
ALTER TABLE T2 MOVE SUBPARTITIONS OF RANGE (a) ((PARTITION 10 <= VALUES < 20) 
    to ('myhost1:30203', 'myhost2:30203');
```

Drop the first-level partition range 5-10 on table 1. The subpartitions for that range only are dropped. The subpartition ranges 10-20 and 20-30 remain.

```
ALTER TABLE T1 DROP PARTITION RANGE (a) (( PARTITION 10 <= VALUES < 15 ));
```

Drop subpartition range 10-100 in range 15-20 on table T1.

```
ALTER TABLE T1 ALTER PARTITION RANGE (a) ((PARTITION 10 <= VALUES < 20)) DROP
    PARTITION RANGE (b) ((PARTITION 10 <= VALUES < 100));
```
Assign the group name GROUP1 to partition 2 on table T1.

```
ALTER TABLE T1 ALTER PARTITION 2 SET GROUP NAME "GROUP1";
```

Assign the group name GROUP2 and the group type HR to the partition range 10 - 20 and all its subpartitions on table T1.

```
ALTER TABLE T1 ALTER PARTITION RANGE (a) ((PARTITION 20 <= VALUES < 30)) SET GROUP NAME "GROUP2" GROUP TYPE "HR" CASCADE;
```

Remove the GROUP value from partition 2 on table T1.

```
ALTER TABLE T1 ALTER PARTITION 2 UNSET GROUP;
```

Remove the GROUP value from the partition range 10 -20 and its subpartitions on table T1.

```
ALTER TABLE T1 ALTER PARTITION RANGE (a) ((PARTITION 20 <= VALUES < 30)) UNSET GROUP CASCADE;
```

This example creates a multilevel heterogeneous partitioned table and then applies the `<load_unit>` attribute first by partition number (1) and then to partition range (10 to 20).

```
CREATE COLUMN TABLE T4 (a INT, b INT) PARTITION BY RANGE(a)     ( ( PARTITION 10 <= VALUES < 20 ) SUBPARTITION BY RANGE (b) ( PARTITION VALUES = 100 ));
ALTER TABLE T4 ALTER PARTITION 1 COLUMN LOADABLE;
ALTER TABLE T4 ALTER PARTITION RANGE (a) ((PARTITION 10 <= VALUES < 20)     SUBPARTITION BY RANGE (b) (PARTITION VALUES = 100)) PAGE LOADABLE;
```

### Dynamic Range Partitioning

These examples are based on table A1 using this CREATE TABLE statement.

```
CREATE COLUMN TABLE A1 (A INT, B INT NOT NULL) PARTITION BY RANGE (A)     ((PARTITION 10 <= VALUES < 20)     SUBPARTITION BY RANGE (B) (PARTITION 15 <= VALUES < 20, PARTITION OTHERS),     (PARTITION VALUES = 30)     SUBPARTITION BY RANGE (B) (PARTITION VALUES=20, PARTITION OTHERS),     (PARTITION OTHERS));
```

Enable dynamic range partitioning with a DYNAMIC interval of 2 years on range 10-20, subpartition 15-20 on table A1.

```
ALTER TABLE A1 PARTITION BY RANGE (A)     ((PARTITION 10 <= VALUES < 20)     SUBPARTITION BY RANGE (B) (PARTITION 15 <= VALUES < 20, PARTITION OTHERS DYNAMIC),     (PARTITION VALUES = 30)     SUBPARTITION BY RANGE (B) (PARTITION VALUES=20, PARTITION OTHERS),     (PARTITION OTHERS));
```

Add another partition from OTHERS or subpartition 15-20 on table A1.

```
ALTER TABLE A1 ADD PARTITION FROM OTHERS;
```

Drop any empty OTHERS partitions on table A1.

```
ALTER TABLE A1 DROP EMPTY PARTITIONS;
```
Set the threshold to 2 for dynamic range partitioning on the subpartition of range 10-20 on table A1.

```
ALTER TABLE A1 PARTITION BY RANGE (A)
    ((PARTITION 10 <= VALUES < 20)
     SUBPARTITION BY RANGE (B) (PARTITION 15 <= VALUES < 20, PARTITION OTHERS DYNAMIC THRESHOLD 2),
     (PARTITION VALUES = 30)
     SUBPARTITION BY RANGE (B) (PARTITION VALUES = 20, PARTITION OTHERS)),
    (PARTITION OTHERS));
```

Disable dynamic range partitioning on the subpartition on range 10-20 on table A1.

```
ALTER TABLE A1 PARTITION BY RANGE (A)
    ((PARTITION 10 <= VALUES < 20)
     SUBPARTITION BY RANGE (B) (PARTITION 15 <= VALUES < 20, PARTITION OTHERS),
     (PARTITION VALUES = 30)
     SUBPARTITION BY RANGE (B) (PARTITION VALUES = 20, PARTITION OTHERS),
     (PARTITION OTHERS));
```

### Related Information

- Non-heterogeneous Alter Partition Clauses [page 630]
- ALTER TABLE Statement (Data Definition) [page 589]
- Table Partitioning
- Dynamic Range Partitioning
- SAP HANA Native Storage Extension
- TABLE_PARTITIONS System View [page 1681]

### 4.10.1.23 ALTER USER Statement (Access Control)

Modifies the database user.

### Syntax

```
ALTER USER <user_name> {  
   <remote_identity_option>  
   [ <password_validation_option> [ <validity_specification> ] [  
       <user_parameter_option> ]  
   | <usergroup_membership_option> 
   [ <validity_specification> [ <user_parameter_option> ] [  
       <user_parameter_option> ]  
   ]  
   | <external_ident> [ <validity_specification> ] [ <user_parameter_option> ]  
   | <reset_connect_attempts>  
   | <drop_connect_attempts>  
   | <password_lifetime>  
   | <force_pass_change>  
   | <user_activation_opts>  
   | <authent_mech_opts>  
```

SAP HANA SQL Reference Guide for SAP HANA Platform
Syntax Elements

user_name
Specifies the username of the user to be modified.

\[<user_name> ::= <unicode_name>\]

remote_identity_option
Defines a remote database identification mechanism.

\[ADD \ REMOTE \ IDENTITY \ <remote_user_name> \ AT \ DATABASE \ <database_name>\]

remote_user_name
Specifies the remote user name.

\[<remote_user_name> ::= <unicode_name>\]

database_name
Specifies the remote database name.

\[<database_name> ::= <identifier>\]

password_validation_option
Specifies the user password.

\[<password_validation_option> ::= PASSWORD <password> \ [ \ NO \ FORCE_FIRST_PASSWORD_CHANGE \ ]\]

The password must follow the rules defined for the current SAP HANA instance. The password rules include a minimal password length and a definition of which character types (lower, upper, digit, special characters) must be part of the password.

NO FORCE_FIRST_PASSWORD_CHANGE overrules the setting of the password policy parameter FORCE_FIRST_PASSWORD_CHANGE and allows the final password to be specified during user creation.

validity_specification
Specifies the validity specification with an optional user parameter option.

\[<validity_specification> ::= VALID <validity_opts>\]

validity_opts
Configures the user’s temporal validity. The specified user is only allowed to connect within the given date range. Using this feature you can create users in advance or restrict the period when users can connect to the SAP HANA database.

```sql
<validity_opts> ::=    <from_specification> [ <until_specification> ]    | <until_specification>
```

**from_specification**
Sets when the user is valid from.

```sql
<from_specification> ::= FROM { <timestamp> | NOW }
```

Use the NOW keyword to configure the user account to be valid from the current time.

The default is NOW.

**until_specification**
Specifies when the user is valid until.

```sql
<until_specification> ::= UNTIL { <timestamp> | FOREVER }
```

Use the FOREVER keyword to configure the user account to never expire.

The default is FOREVER.

**timestamp**
Specifies a timestamp.

```sql
<timestamp> ::= <string_literal>
```

**user_parameter_option**
Sets or clears user parameters.

In the case of competing settings between user parameter options and workload class settings, a workload class setting takes precedence over a user parameter option. Workload class settings and user parameter options always take precedence over .ini file settings.

```sql
[user_parameter_option] ::=    <set_user_parameters> [ <clear_user_parameter_option> ]   | <clear_user_parameter_option>
```

**set_user_parameters**
Sets parameters in the user parameter list.

```sql
<set_user_parameters> ::= SET PARAMETER <user_parameter_list>
```

**user_parameter_list**
Specifies a list of user parameters.

```sql
[user_parameter_list] ::=    <user_parameter> [, <user_parameter> [,...] ]
```

```sql
[user_parameter] ::= CLIENT = <string_literal>
```
<table>
<thead>
<tr>
<th>User Parameter</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT</td>
<td>When you define column store views, this parameter restricts the user’s access to the specified client. This parameter cannot be specified by users themselves.</td>
</tr>
<tr>
<td>LOCALE</td>
<td>When you define column store views, the user parameter LOCALE translates information according to the user’s locale.</td>
</tr>
<tr>
<td>TIME ZONE</td>
<td>This parameter is not used by the SAP HANA database, but it can be read by external applications.</td>
</tr>
<tr>
<td>EMAIL ADDRESS</td>
<td>This parameter is not used by the SAP HANA database, but can be read by external applications. This value must be unique.</td>
</tr>
<tr>
<td>RSERVE REMOTE SOURCES</td>
<td>For use with the rserve adapter, this parameter specifies rserve remote sources.</td>
</tr>
<tr>
<td>STATEMENT MEMORY LIMIT</td>
<td>Sets a user-specific statement memory limit in gigabytes.</td>
</tr>
<tr>
<td></td>
<td>• If both a global and a user statement memory limit are set, then the user specific limit takes precedence, regardless of whether it is higher or lower than the global statement memory limit.</td>
</tr>
<tr>
<td></td>
<td>• If the user-specific statement memory limit is removed, then the global limit takes effect for the user.</td>
</tr>
<tr>
<td></td>
<td>• Setting the statement memory limit to 0 disables any statement memory limit for the user.</td>
</tr>
<tr>
<td></td>
<td>• For STATEMENT MEMORY LIMIT to take effect, resource_tracking and memory_tracking must be active.</td>
</tr>
<tr>
<td>STATEMENT THREAD LIMIT</td>
<td>Sets a user specific concurrency limit on statements (despite the name, STATEMENT THREAD LIMIT is not an actual thread limit). Similar behaviors to STATEMENT MEMORY LIMIT apply for STATEMENT THREAD LIMIT.</td>
</tr>
</tbody>
</table>
### User Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARAMETER PRIORITY</td>
<td>Sets a user-level priority value for all statements in the current connection; the range of possible values is from 0 to 9 (the default is 5).</td>
</tr>
</tbody>
</table>

#### clear_user_parameter_option

Clears one or more user parameters.

```
<clear_user_parameter_option> ::=  
  CLEAR PARAMETER <clear_user_parameter_list>  
  | CLEAR ALL PARAMETERS  
<clear_user_parameter_list> ::=  
  <clear_user_parameter> [, <clear_user_parameter> [...]]
```

For the list of parameters you can clear, refer to the list you can set in the `<user_parameter_list>` in the previous clause.

#### usergroup_membership_option

Adds or removes a user from a user group. A user can only be a member of one user group at a time.

```
<usergroup_membership_option> ::=  
  SET USERGROUP <usergroup_name>  
  | UNSET USERGROUP
```

**SET USERGROUP usergroup_name**

Adds the user to the specified user group.

**UNSET USERGROUP**

Removes the user from the specified user group.

When adding and removing a user to/from a user group, you must have the appropriate privileges for the user group you are adding to and removing from, as follows:

- For user groups defined as ENABLE USER ADMIN, you must have USER ADMIN privilege, or USERGROUP OPERATOR privilege on the user group, to add/remove users.
- For user groups defined as DISABLE USER ADMIN, you must have USERGROUP OPERATOR privilege on the user group to add/remove users.

#### external_ident

Specifies the external identity for the user. This clause is only for Kerberos authentication.

```
IDENTIFIED EXTERNALLY AS <external_identity> [ VALID <validity_specification> ]
```

**external_identity**

Defines an external identity that authenticates the user.

```
<external_identity> ::=  
  <simple_identifier>  
  | <string_literal>
```
Resets the number of invalid connection attempts to zero so the user can connect immediately.

\[
<\text{reset\_connect\_attempts}> ::= \text{RESET CONNECT ATTEMPTS}
\]

You need the USER ADMIN privilege to use this command.

For the maximum number of allowed invalid connection attempts before a user is locked, see the M\_PASSWORD\_POLICY system view.

\text{drop\_connect\_attempts}

Drops old information about invalid connect attempts recorded for the specified user.

\[
<\text{drop\_connect\_attempts}> ::= \text{DROP CONNECT ATTEMPTS}
\]

It does not reset the current count of invalid connect attempts and therefore does not allow the user to connect immediately.

This option can be used by the user themselves or by a user with the USER ADMIN privilege.

To view the count of invalid connection attempts that have occurred, see the INVALID\_CONNECT\_ATTEMPTS system view.

\text{password\_lifetime}

Enables or disables password lifetime checks.

\[
<\text{password\_lifetime}> ::= \{ \text{ENABLE} \mid \text{DISABLE} \} \text{ PASSWORD LIFETIME}
\]

You need the USER ADMIN privilege to use this command.

\text{force\_pass\_change}

Forces the user to change their password immediately after the next connection to the SAP HANA database.

\[
<\text{force\_pass\_change}> ::= \text{FORCE PASSWORD CHANGE}
\]

You need the USER ADMIN privilege to use this command.

\text{user\_activation\_opts}

Activates or deactivates a user’s account.

\[
<\text{user\_activation\_opts}> ::= \{ \text{ACTIVATE} \mid \text{DEACTIVATE} \} [ \text{USER NOW} ]
\]

Users cannot connect to the SAP HANA database once their account is deactivated. However, it may appear as though deactivated users are still active in the system (for example when a procedure that was created by the user with DEFINER MODE is called). Reactivate a deactivated user with the ACTIVATE keyword. Activating and reactivating a user requires the USER ADMIN privilege.

\text{authent\_mech\_opts}

Enables or disables the selected authentication mechanism.

\[
<\text{authent\_mech\_opts}> ::= \{ \text{ENABLE} \mid \text{DISABLE} \} <\text{authentication\_mechanism}>
\]

\text{authentication\_mechanism}

Specifies the type of the authentication mechanism to enable or disable.

\[
<\text{authentication\_mechanism}> ::= 
\]
### add_ident_opts

Defines a method for authenticating the user.

```
<add_ident_opts> ::=  
ADD IDENTITY { <add_provider_identity> [ <add_provider_identity> [...] ]  
| { <external_identity> FOR KERBEROS }  
<add_provider_identity> ::=  
<saml_provider_ident>  
|x509_provider_ident>  
| <kerberos_provider_ident>  
| <logon_ticket_ident>  
| <assertion_ticket_ident>  
| <jwt_provider_ident>  
```

### saml_provider_ident

Defines a SAML provider.

```
<saml_provider_ident> = <mapped_user_name> FOR SAML PROVIDER 
<saml_provider_name>  
<mapped_user_name> ::= ANY | <string_literal>  
<saml_provider_name> ::= <simple_identifier> 
```

- `<mapped_user_name>` specifies the mapped SAML user name to use. If the keyword ANY is used, then the SAML assertion contains the name of the database user that the assertion is valid for.

- `<saml_provider_name>` specifies the identifier of a SAML provider that already exists in the system.

For more information on SAML providers, see the CREATE SAML PROVIDER statement.

### x509_provider_ident

Defines an X.509 certificate issuer or X.509 provider.

```
<x509_provider_ident> ::=  
<subject_distinguished_name> ISSUER <issuer_distinguished_name> FOR X509  
| { <subject_distinguished_name> | ANY } FOR X509 PROVIDER 
<x509_provider_name>  
<subject_distinguished_name> ::= <string_literal>  
<issuer_distinguished_name> ::= <string_literal>  
<x509_provider_name> ::= <simple_identifier> 
```

- `<subject_distinguished_name>` specifies the subject name provided in the certificate of the X.509 identity provider.

- `<issuer_distinguished_name>` specifies the issuer name provided in the certificate of the X.509 identity provider.

You can specify a wildcard (ANY) for the subject name only for an X.509 provider with matching rules.

- `<x509_provider_name>` specifies the identifier of an X.509 provider that already exists in the system.

### kerberos_provider_ident
Defines a KERBEROS identity.

```
<kerberos_provider_ident> ::= <kerberos_principal_name> FOR KERBEROS
<kerberos_principal_name> ::= <string_literal>
```

*<kerberos_principal_name>* specifies an identity within an external authentication system.

### logon_ticket_ident

Sets SAP Logon Ticket as the authentication method.

```
<logon_ticket_ident> ::= FOR SAP LOGON TICKET
```

### assertion_ticket_ident

Sets SAP Assertion Ticket as the authentication method.

```
<assertion_ticket_ident> ::= FOR SAP ASSERTION TICKET
```

### jwt_provider_ident

Defines a JWT provider-user-mapping.

```
<jwt_provider_ident> ::= <mapped_user_name> FOR JWT PROVIDER
<jwt_provider_name> <mapped_user_name> ::= { ANY | <string_literal> }
<jwt_provider_name> ::= <simple_identifier>
```

*<mapped_user_name>* specifies the mapped JWT user name to use. If the keyword ANY is used, then the JWT token contains the name of the database user that the token is valid for.

*<jwt_provider_name>* specifies the identifier of a JWT provider already existing in the system.

### drop_ident_opts

Drops the defined identity provider.

```
<drop_ident_opts> =
DROP IDENTITY { <provider_info> [ <provider_info> [...] ] | FOR KERBEROS }
```

#### provider_info

Specifies the provider info to be dropped.

```
<provider_info> ::= [ <mapped_user_name> ] FOR SAML PROVIDER <saml_provider_name>
| [ 'subject_distinguished_name' ISSUER 'issuer_distinguished_name' ] FOR X509
| [ 'subject_distinguished_name' ] | ANY ] FOR X509 PROVIDER
```

Specifying a subject and issuer for an X.509 provider removes only that mapping. Specifying ANY removes the wildcard mapping. If you don’t specify a subject and issuer or ANY, all mappings between the user and provider are removed.

Specifying a mapped user name or ANY for a JWT or SAML provider removes only that mapping. If you don’t specify a mapped user name, all mappings between the user and provider are removed.

### add_remote_database_identity

SAP HANA SQL Reference Guide for SAP HANA Platform
Adds a remote database identity.

```sql
ADD REMOTE IDENTITY <remote_user_name> AT DATABASE <database_name>
```

**drop_remote_database_identity**

Drops a remote database identity.

```sql
DROP REMOTE IDENTITY <remote_user_name> AT DATABASE <database_name>
```

**client_connect_option**

Enables or disables the client from connecting at all. In case of disabled client connect, only connecting via an XS application is allowed.

```sql
<client_connect_option> ::= { ENABLE | DISABLE } CLIENT CONNECT
```

**special_privilege_handling**

Allows a user with the USER ADMIN privilege to control whether another user can create objects in their own schema, and whether the user has the PUBLIC role.

When a user is created, they are automatically granted PUBLIC and CREATE ANY for objects in their schema, and these are granted by user SYS. `<special_privilege_handling>` allows users with the USER ADMIN privilege to remove (or re-grant) CREATE ANY and PUBLIC. Users that do not have the PUBLIC role and do not have the CREATE ANY ON OWN SCHEMA privilege are indicated in the USERS system view as restricted users.

```
<special_privilege_handling> ::=  { GRANT | REVOKE } CREATE ANY ON OWN SCHEMA
    | { GRANT | REVOKE } ROLE PUBLIC
```

Use the GRANT | REVOKE CREATE ANY ON OWN SCHEMA clause to grant or revoke, respectively, a user’s ability to create objects in their own schema.

Use the GRANT | REVOKE ROLE PUBLIC clause to grant or revoke, respectively, PUBLIC role from a user.

**ldap_group_authorization**

Specifies the LDAP group authorization mode for the user.

```
<ldap_group_authorization> ::= AUTHORIZATION { LOCAL | LDAP }
```

**LOCAL**

Specifies the use of local authorization mode for the user. This is the default.

**LDAP**

Specifies the use of LDAP authorization mode for the user.

Users with LDAP authorization mode cannot be granted permissions directly and cannot have local roles granted to them.

When changing the LDAP authorization mode from LOCAL authorization to LDAP for a user, LDAP roles are evaluated and activated upon next login. Also, local roles and privileges granted to the user are revoked except for roles that are granted by SYS—for example, the PUBLIC role—and the CREATE ANY privilege on the user’s own schema.
Description

Users configured for LDAP authentication cannot simultaneously be configured for local password authentication. Local password authentication must be disabled before a user can be configured for LDAP authentication, for example, using ALTER USER...DISABLE PASSWORD.

External users are authenticated by using an external system, for example a Kerberos system. For detailed information about external identities, contact your domain administrator.

Permissions

Every database user has the required privileges to call the ALTER USER statement. However, some of the options for this command required additional database privileges.

Administrators can use the ALTER USER statement to configure other user accounts as long as they have the USER ADMIN privilege.

Configuration Parameters

Configuration parameters concerning the user’s password can be observed with the M_PASSWORD_POLICY system view. These parameters are stored in indexserver.ini, in the password policy section.

The description of the parameters concerned can be found in the Appendix of the SAP HANA Security Guide under Password Policy Parameters.

Examples

Create a SAML provider named ac_saml_provider in the database specifying a subject and issuer for ACompany.

```sql
CREATE SAML PROVIDER ac_saml_provider WITH SUBJECT 'CN = wiki.detroit.ACompany.corp,OU = ACNet,O = ACompany,C = EN' ISSUER 'E = John.Do@acompany.com,CN = ACNetCA,OU = ACNet,O = ACompany,C = EN';
```

Create a new user named new_user that can connect by using a password or with an assertion of the SAML provider ac_saml_provider. The <mapped_user_name> was set to ANY as the assertion provides the database user name.

```sql
CREATE USER new_user PASSWORD Password1 WITH IDENTITY ANY FOR SAML PROVIDER ac_saml_provider;
```

Force the user to change their password.

```sql
ALTER USER new_user FORCE PASSWORD CHANGE;
```
Disable SAML authentication for the user.

```
ALTER USER new_user DISABLE SAML;
```

Reset the number of invalid connection attempts to zero for new_user.

```
ALTER USER new_user RESET CONNECT ATTEMPTS;
```

Define the external identity for KERBEROS and enable KERBEROS for this user. Adding an external identification mechanism to a user does not automatically enable it. You must do this as a separate step as shown in this example.

```
ALTER USER new_user ADD IDENTITY 'testkerberosName' FOR KERBEROS;
ALTER USER new_user ENABLE KERBEROS;
```

Remove the ac_saml_provider SAML provider identity from new_user.

```
ALTER USER new_user DROP IDENTITY FOR SAML PROVIDER ac_saml_provider;
```

Disable the account.

```
ALTER USER new_user DEACTIVATE;
```

**Related Information**

- CREATE USER Statement (Access Control) [page 893]
- USERGROUPS System View [page 1696]
- USERS System View [page 1698]
- SQL Notation Conventions [page 30]
- Data Types [page 33]
- M_PASSWORD_POLICY System View [page 2039]
- INVALID_CONNECT_ATTEMPTS System View [page 1588]
- CREATE SAML PROVIDER Statement (Access Control) [page 803]
- USER_PARAMETERS System View [page 1702]
- SAML_PROVIDERS System View [page 1648]
- SAML_USER_MAPPINGS System View [page 1649]
4.10.1.24 ALTER USERGROUP Statement (Access Control)

Alters a usergroup.

Syntax

```
ALTER USERGROUP <usergroup_name>
{ ENABLE | DISABLE } USER ADMIN
[ SET PARAMETER <parameter_key_value_list> ]
[ CLEAR PARAMETER <parameter_name_list> ]
[ { ENABLE | DISABLE } PARAMETER SET <parameter_set_name> ]
[ { ENABLE | DISABLE } CLIENT CONNECT ]
```

Syntax Elements

`<usergroup_name>`

Specifies the name of the usergroup.

{ ENABLE | DISABLE } USER ADMIN

Controls whether a user with USER ADMIN privilege can administer the usergroup (change its settings, and add/remove users).

Users with USERGROUP OPERATOR privilege on the usergroup can always administer the usergroup. ENABLE USER ADMIN allows any user with USER ADMIN privilege to administer the group in addition to users with USERGROUP OPERATOR privilege on the usergroup.

DISABLE USER ADMIN allows only users with the USERGROUP OPERATOR privilege on the usergroup to administer the group.

SET PARAMETER `parameter_key_value_list`

Configures parameter options as a key-value list for a parameter set that can be enabled and disabled for the group. Specifically, use this clause to configure group-specific values for the individual options of the password policy set called ‘password policy’.

```
<parameter_key_value_list> ::= <key_value_pair> [,<key_value_pair> [,...]]
<key_value_pair> ::= '<parameter_name>'='<value>'
```

When the SET PARAMETER clause is used to specify password policy options, all policy option settings are stored for the usergroup with the exception of password_lock_for_system_user, which is not applicable or allowed for usergroups. However, this does not mean you need to specify all of the password policy options. Password policy options that are not specified in the SET PARAMETER clause are copied from the current system-wide password policy settings and stored with the specified options for the usergroup. Subsequent changes to the system-wide settings do not impact any stored password policy options for the usergroup. Refer to the SAP HANA Administration Guide for the list of password policy options you can set.

Once you set the parameter set options, use an ALTER USERGROUP...SET PARAMETER statement to alter specific parameter settings without impacting other parameter settings that have been set.

CLEAR PARAMETER `parameter_name_list`
Resets the specified parameter(s) to the current system-wide setting.

```plaintext
<parameter_name_list> ::= <parameter_name> [, <parameter_name> [,..] ]
```

**ENABLE PARAMETER SET parameter_set_name**

Enables the specified parameter set. The only supported parameter set name is ‘password policy’.

```plaintext
<parameter_set_name> ::= 'password policy'
```

Enabling the password policy parameter causes any password policy options that have been defined to be applied. If ENABLE PARAMETER SET ‘password policy’ is not specified, any password policy options configured using the SET PARAMETER clause remain configured for the usergroup, but not applied. Also, if a parameter set is enabled before any parameters have been set, all parameter values are identical to the current system wide settings.

**DISABLE PARAMETER SET parameter_set_name**

Disables the specified parameter set. Parameters that have been set remain stored for the usergroup but are not applied. The only supported parameter set name is ‘password policy’.

```plaintext
<parameter_set_name> ::= 'password policy'
```

**ENABLE CLIENT CONNECT** Allows users in the usergroup to connect to the database.

**DISABLE CLIENT CONNECT** Prevents users in the usergroup from connecting to the database. You cannot disable client connections to the usergroup you are a member of.

**Description**

To see the group-specific values of parameters, query the USERGROUP_PARAMETERS system view. To see which values are currently in effect for a particular user, query the M_EFFECTIVE_PASSWORD_POLICY system view.

You add users to a usergroup using the CREATE USER statement.

The M_EFFECTIVE_PASSWORD_POLICY system view can be used to query the current password policy settings for a user.

**Permissions**

Only users with USER ADMIN privilege can specify ENABLE/DISABLE USER ADMIN.

For other changes, if the current setting for the usergroup is DISABLE USER ADMIN, then only users with the USERGROUP OPERATOR privilege on the usergroup can alter the group. Otherwise, any user with the USER ADMIN privilege can alter the usergroup, in addition to any user with the USERGROUP OPERATOR privilege on the usergroup.
Example

The following statement alters the usergroup MyUserGroup to allow any user with USER ADMIN privilege to administer it:

```
ALTER USERGROUP MyUserGroup ENABLE USER ADMIN;
```

The following example disables the password policy parameter set for the usergroup MyUserGroup:

```
ALTER USERGROUP MyUserGroup DISABLE PARAMETER SET 'password policy';
```

The following example changes MyUserGroup to allow connections to the database by members of the group:

```
ALTER USERGROUP MyUserGroup ENABLE CLIENT CONNECT;
```

This example changes MyUserGroup to restrict connections to the database by members of the usergroup:

```
ALTER USERGROUP MyUserGroup DISABLE CLIENT CONNECT;
```

Related Information

User Groups
Password Policy Configuration Options
CREATE USERGROUP Statement (Access Control) [page 902]
DROP USERGROUP Statement (Access Control) [page 987]
GRANT Statement (Access Control) [page 1010]
USERGROUPS System View [page 1696]
USERS System View [page 1698]
CREATE USER Statement (Access Control) [page 893]
USERGROUP_PARAMETERS System View [page 1697]

4.10.1.25  ALTER VIEW Statement (Data Definition)

Alters the definition, restrictions, or options on a view.

Syntax

```
ALTER VIEW <view_name> (  
  ( <column_name_list> ) ] AS <subquery> [ [ <view_option_list> ] [ <view_cache> ] ] <cache_clause> <mask_clause> <expression_macros> <view_access_modes>
```

SAP HANA SQL Reference Guide for SAP HANA Platform
Syntax Elements

view_name

Creates the specified view, with an optional schema name.

\[ <view_name> ::= [ <schema_name> \.]<identifier> \]
\[ <schema_name> ::= <unicode_name> \]

column_list

Specifies the column names for the view.

\[ <column_list> ::= <column_name> [, <column_name> [,…] ] \]

When a column name is specified along with the view name, a query result is displayed with that column name. If a column name is omitted, then a query result gives an appropriate name to the column automatically. The number of column names must be the same as the number of columns returned from <subquery>.

AS subquery Specifies the subquery to use for populating the view with data.

view_restrictions

Sets restrictions based on structured privileges.

\[ <view_restriction> ::= WITH CHECK OPTION [ <column_mask_clause> ] [ <structured_privilege_check> ] | WITH NO CHECK OPTION [ <column_mask_clause> ] [ <structured_privilege_check> ] | WITH READ ONLY [ <column_mask_clause> ] [ <structured_privilege_check> ] | WITH ... ::= <column_name> USING <mask_expression>  <structured_privilege_check> ::= { STRUCTURED PRIVILEGE CHECK | NO STRUCTURED PRIVILEGE CHECK } \]

<mask_expression> can be any type of expression, including a user-defined function, but it cannot be a subquery.

For more information on data masking, see the SAP HANA Security Guide.

with_association_clause

Defines a relationship between the view and one or more tables (or views, in the case of JOIN).

\[ <with_association_clause> ::= WITH ASSOCIATIONS { <association_def_list> } <association_def_list> ::= <association_def> [,<association_def> [,…] ] <association_def> ::= ( <join_definition> | <column_def> ) <join_definition> ::= JOIN <table_or_view_identifier> [ AS <table_alias> ] ON { <condition> | <column_def> } \]
with_annotation_clause

Alters table-, column-, and parameter-level annotations. You can reference annotations in subsequent queries to filter results.

While you must specify annotation on at least one object with the WITH ANNOTATION clause (view, column, or parameter), there is no limit on the number of annotations or types of annotations you can specify.

<key> and <value> represent the annotations you are configuring for the object (view, column, or parameter).

cache_clause

Changes result cache settings for a view.

add_cache_clause

Caches the result of a table function with an optional projection list and filter conditions. A result cache can improve performance for subsequent queries on the view.
Specify whether your result cache is static or dynamic.

```plaintext
<cache_type> ::= STATIC | DYNAMIC
```

A view has only one type of result cache. The default value is STATIC. Using a static result cache may result in stale data. A dynamic result cache ensures up-to-date data as it is partially or fully refreshed every time you run a query on the result cache.

**minute_value**

Specifies the result cache retention period.

```plaintext
<minute_value> ::= <unsigned_integer>
```

Specifying a retention period ensures that data does not exceed the specified RETENTION period. Data that exceeds the retention period is refreshed and up-to-date data is returned. Since a dynamic result cache ensures up-to-date data, specifying a retention period of any value other than zero (0) generates an error. Dynamic result caches are only supported for SQL views that are defined as an aggregation on single-column tables.

**projection_list**

For static result caches, allows you to reduce the cached data by specifying a result cached projection list.

```plaintext
<projection_list> ::= <projection_name> [{, <projection_name>}]  
<projection_name> ::= <column_name> | <aggr_type>(<column_name>)  
<aggr_type> ::= { SUM | MIN | MAX | COUNT | AVG }
```

If a column that is not part of the projection list is requested or included in the WHERE clause, then the result cache cannot be exploited. You can also direct an aggregation type of a specific column for the SQL view (not supported for calculation view). You specify a result cached projection list, and then the result cache includes aggregated results of that column and returns aggregated results only. This option is not supported for dynamic result caches.

**filter_condition**

Reduces the result cached data by specifying a filter condition.

```plaintext
<filter_condition> ::=  
<condition> OR <condition>  
| <condition> AND <condition>  
NOT <condition>  
{ <condition> }  
<predicate>
```

```plaintext
<predicate> ::=  
<comparison_predicate>  
| <range_predicate>  
| <like_predicate>  
| <null_predicate>
```

```plaintext
<comparison_predicate> ::=  
<table_expression> { = | != | <> | > | < | >= | <= | ANY | SOME | ALL } ( <table_expression_list> )
```

```plaintext
<range_predicate> ::=  
<table_expression> [ NOT ] BETWEEN <table_expression> AND <table_expression>
```

```plaintext
<in_predicate> ::=  
```
Only filtered results are cached. Predicates that contain subqueries are not supported.

For static result caches, all forms of non-parameterized query filters are supported. For parameterized query filters, only conjunctive forms are supported. For dynamic result caches, only parameterized query filters on string type columns in conjunctive forms are supported. For example:

```sql
column = ? [ AND <and_column = ?> ]
```

<location_clause>
Specifies the location where the result cache entry is created.

```sql
<location_clause> ::= AT [LOCATION] '<hostname>:<port_number>'[, ...]
```

By default, when you execute a query on a static or dynamic result cached view, the optimizer determines the result cache location. In some cases, the same result cache entry is created at multiple indexservers.

<alter_cache_clause>
Changes the result cache.

```sql
<alter_cache_clause> ::= ALTER [ <cache_type> ] CACHE
   { <alter_retention_clause> | <alter_location_clause> | <index_clause> }
<cache_type> ::= STATIC | DYNAMIC
<alter_retention_clause> ::= RETENTION <minute_value>
<alter_location_clause> ::= { ADD | DROP } AT [ LOCATION ]
'='<hostname>:<port_number>'[, ...]
<index_clause> ::= { ADD | DROP } INDEX ON <column_name>
```

For static caches, you can change the retention value of the result cache of a view. For dynamic caches, only the value 0 is supported.

Use the <location_clause> to change the location where the cache entry is created. By default, when you execute a query on a static or dynamic cached view, the optimizer determines the cache location. In some cases, the same cache entry is created at multiple indexservers. <host_port> specifies the hostname and port of an currently running indexserver.

Use the <index_clause> to create an index on a dynamic result cache to improve filter evaluation performance on the top of the dynamic result cache. The <index_clause> is not supported for static caches. Dynamic result cache indexes are restricted to single-node systems only.

<drop_cache_clause>
Specifies to drop the result cache of a view.

```sql
<drop_cache_clause> ::= DROP [ <cache_type> ] CACHE
<cache_type> ::= ( STATIC | DYNAMIC )
```
mask_clause

Adds new masking expression or drops an existing mask expression. `<mask_clause>` does not change the view's column definitions. For an example of data masking, see the CREATE VIEW statement.

```
<mask_clause> ::=   { ADD | ALTER } MASK (<column_mask_list>)   | DROP MASK ( <column_name_list> )  <column_mask_list> ... USING <mask_expression> [,…] )  <mask_expression> ::= <expression>  <column_name_list> ::= <expression> AS <expression_macro_alias>  <expression_macro_alias_list> ::= <expression_macro_alias> [,…] 
```

When adding a mask expression, the mask behavior is defined for the object. New column masks must use the same mask mode. When altering, you can change the mask expression, but not the mode. Once all existing masks are dropped, you can add new masks using a different mask mode.

Masking behavior is supported on row and column tables, and SQL row and calculation views. It is not supported on any other type of tables (virtual tables, extended tables, temporary tables etc.) and views (Join/Olap/Hierarchy views).

Only one masking behavior, definer owner-based (DEFAULT MASK) or session user based masking (SESSION USER MASK), is supported on a table or view with masked columns. You can combine both masking behaviors in an object hierarchy. For example, a view with session user based masking can be created on a table with owner-based masking.

If not specified, DEFAULT is the default.

 `<mask_expression>` can be any type of expression, including a user-defined function, that returns the same data type and length as the original column.

For more information on data masking, see the SAP HANA Security Guide for SAP HANA Platform.

expression_macros

Adds or drops one or more expression macros for the view.

Expression macros perform calculations on the results of a query on a view before the results are returned, and are called using the EXPRESSION_MACRO function.

```
<expression_macros> ::=   ADD EXPRESSION MACROS( <expression_macro_list> )   | DROP EXPRESSION MACROS( <expression_macro_alias_list> )  <expression> ::= <expression> AS <expression_macro_alias>  <expression_macro_alias_list> ::= <expression_macro_alias> [,…] 
```

 `<expression>` can be any calculation or aggregation function expression on one or more columns in the query, including a call to the EXPRESSION_MACRO function. A couple examples for `<expression>` might be `AVG(columnA)`, or `EXPRESSION_MACRO(sum_colA)` where `sum_colA` is another expression macro.

 `<expression_macro_alias>` is the name given to the expression macro. You cannot drop an expression macro if another expression macro references it; you must first drop the referring expression macro.

view_access_modes

Specifies the access mode without recompiling the view query or propagating recompilations of dependent view.

```
$view_access_modes> ::= 
```
[NO] READ ONLY Specifies whether the view is read only. If READ ONLY is specified, and one of the base objects is not updatable, then the view cannot be updated. If subsequently the base objects are changed to updatable, and the view is not READ ONLY, then the view becomes updatable without executing the ALTER VIEW statement again.

[NO] DDL ONLY
Controls whether users can query the view or update the underlying table. You can include views that are defined WITH DDL ONLY in other object definitions. Specifying WITH DDL ONLY prevents users from querying the view and from modifying the underlying table.

WITH ANONYMIZATION clause
Specifies data anonymization parameters for the view and its columns. You must execute a REFRESH VIEW statement after configuring anonymization parameters.

```
WITH ANONYMIZATION ( ALGORITHM <algorithm_name> { [ <view_level_parameters> ] } [ <column_level_parameters> ] )
```

algorithm_name
Specifies the algorithm to use to anonymize view data.

```
<algorithm_name> ::= { 'K-ANONYMITY' | 'DIFFERENTIAL_PRIVACY' | 'L-DIVERSITY' }
```

view_level_parameters
Specifies view-level anonymization parameters.

```
/view_level_parameters> ::= PARAMETERS <embedded_hierarchy_expression>
```

<embedded_hierarchy_expression> is a string constant containing view-specific anonymization parameters.

column_level_parameters
Specifies column-level anonymization parameters.

```
/column_level_parameters> ::= <column_spec> [ <column_spec> [... ] ]
/column_spec> ::= COLUMN <column_name> PARAMETERS <embedded_hierarchy_expression>
```

<embedded_hierarchy_expression> is a string constant containing column-specific anonymization parameters.

<column_name> must be the name of a column previously specified in <column_name_list>.

For more information, see the topics on data anonymization in the SAP HANA Administration Guide.

Description

Alters the definition, restrictions, or options on a view.
To change the comment on a view, use the COMMENT ON statement.
Examples

Result caching examples

The following example creates a table named TAB, and then creates a view, V, that selects all records from it:

```
CREATE COLUMN TABLE TAB (COL1 INT PRIMARY KEY, COL2 INT);
CREATE VIEW V AS SELECT DISTINCT COL1, COL2 FROM TAB;
```

The following example adds the result cache to view V, and enable static result caching.

```
ALTER VIEW V ADD STATIC CACHE RETENTION 10;
```

The following example changes the retention value of the result cache of the view V.

```
ALTER VIEW V ALTER STATIC CACHE RETENTION 20;
```

The following example drops the result cache of view V.

```
ALTER VIEW V DROP STATIC CACHE;
```

The following example enables dynamic result caching on view V.

```
ALTER V ADD DYNAMIC CACHE;
```

The following example restricts the cache entry location for view_a to the location myhost2:00002.

```
ALTER VIEW view_a ADD STATIC CACHE AT LOCATION 'myhost2:00002';
```

The following example removes the location specifications so that there are no restrictions on the cache entry location for view_a.

```
ALTER VIEW view_a ALTER STATIC CACHE DROP AT LOCATION 'myhost2:00002';
```

The following statements show how you can add, alter, or drop the RESULT CACHE on a view:

```
CREATE ROW TABLE TAB (COL1 INT PRIMARY KEY, COL2 INT);
CREATE VIEW V AS SELECT * FROM TAB;
ALTER VIEW V ADD STATIC CACHE RETENTION 10;
ALTER VIEW V ALTER STATIC CACHE RETENTION 20;
ALTER VIEW V DROP STATIC CACHE;
```

Static cache retention example

Create and populate table T and then create view V_1 based on the table:

```
CREATE COLUMN TABLE T (A INT, B INT, C INT);
INSERT INTO T VALUES (1, 1, 10);
INSERT INTO T VALUES (2, 2, 10);
INSERT INTO T VALUES (3, 3, 10);
CREATE VIEW V_1 AS (SELECT A, B, C FROM T);
```

Then alter view V_1 to add to add a static cache retention of 1000 minutes for column A, the aggregated sum of column B, and the aggregated average of column C:

```
ALTER VIEW V_1 ADD STATIC CACHE RETENTION 1000 OF A, SUM(B), AVG(C);
```
When you run the following SELECT statement, the new cache is used:

```sql
SELECT A, SUM(B), AVG(C) FROM V_1 GROUP BY A WITH HINT (RESULT_CACHE);
```

**Add/drop dynamic cache index examples**

```sql
CREATE COLUMN TABLE tab_a ( A INT, B INT, C INT, D INT);
CREATE VIEW view_a AS (SELECT A, B, sum(D) D from tab_a group by A, B);
ALTER VIEW view_a ADD DYNAMIC CACHE;
```

Execute the following statement to add a dynamic cache index on the column A:

```sql
ALTER VIEW view_a ALTER DYNAMIC CACHE ADD INDEX ON A;
```

Execute the following statement to add an additional dynamic cache index on the column B:

```sql
ALTER VIEW view_a ALTER DYNAMIC CACHE ADD INDEX ON B;
```

Execute the following statement to drop the dynamic cache index on column B:

```sql
ALTER VIEW view_a ALTER DYNAMIC CACHE DROP INDEX ON B;
```

**Expression macros examples**

The following statements create a view with two expression macros (sum_a and count_a) and then alters the view to add another expression macro (avg_a) to it:

```sql
CREATE TABLE t1(a INT);
CREATE VIEW v1 AS SELECT * FROM t1 WITH EXPRESSION MACROS(SUM(a) AS sum_a,
COUNT(a) AS count_a);
ALTER VIEW v1 ADD EXPRESSION MACROS(AVG(a) AS avg_a);
```

The following statement alters the view to add an expression macro (avg2_a) that also calculates the average of column `a` but by using two of the other expression macros in the calculation:

```sql
ALTER VIEW v1 ADD EXPRESSION MACROS(EXPRESSION_MACRO(sum_a) /
EXPRESSION_MACRO(count_a) AS avg2_a);
```

The following example drops the two expression macros `avg_a` and `avg2_a`:

```sql
ALTER VIEW v1 DROP EXPRESSION MACROS( avg_a, avg2_a );
```

**WITH ANNOTATIONS examples**

The following statement alters annotations on the myView view:

```sql
ALTER VIEW myView WITH ANNOTATIONS (  UNSET 'key1' SET 'key1' = 'value1'  COLUMN column1 UNSET 'Key2' SET 'Key2' = 'value2' );
```

**WITH DDL ONLY examples**

The following statements show how to use the WITH DDL ONLY clause to control whether users can query a view:

```sql
CREATE TABLE t1 (a int, b int);
CREATE VIEW v1 AS SELECT * FROM t1 WITH DDL ONLY;
ALTER VIEW v1 AS SELECT * FROM t1 WITH NO DDL ONLY;
```
Related Information

CREATE VIEW Statement (Data Definition) [page 904]
VIEW_EXPRESSION_MACROS System View [page 1708]
EXPRESSION_MACRO Function (Miscellaneous) [page 193]
COMMENT ON Statement (Data Definition) [page 718]
ANNOTATE Statement (Data Definition) [page 686]
ANNOTATIONS System View [page 1494]
REFRESH VIEW Statement (Data Definition) [page 1086]
SAP HANA Data Anonymization
Predicates [page 61]

4.10.1.26 ALTER VIRTUAL TABLE Statement (Data Definition)

Modifies a virtual table’s column properties, and lets you refresh the metadata of a virtual table.

Syntax

ALTER VIRTUAL TABLE <virtual_table_name>   { SET PROPERTY 'name' = 'value'[, 'name2' = 'value2'[, ...] ]
   | UNSET PROPERTY 'name'[, 'name2', ...]
   | ALTER <column_name> SET PROPERTY 'name' = 'value' [, 'name2' = 'value2'
   [, ...] ]
   | ALTER <column_name> UNSET PROPERTY 'name'[, 'name2'[, ...] ]
   | REFRESH DEFINITION }

Syntax Elements

virtual_table_name

Specifies a virtual table.

<virtual_table_name> ::= [ <schema_name>.]<identifier>

column_name
Specifies a virtual column.

<column_name> ::= <identifier>

**REFRESH DEFINITION** Updates a virtual table to reflect metadata changes in the corresponding remote table.

**Description**

Any properties that are not set by using the ALTER VIRTUAL TABLE statement are marked as read-only, which cannot be updated by the ALTER VIRTUAL TABLE statement.

Use the DROP TABLE `<table_name>` to drop a virtual table.

**Permissions**

This statement requires the CREATE VIRTUAL TABLE object privilege.

**Examples**

Set a new property for the virtual table REMOTE1_VT, then update and unset it.

```
ALTER VIRTUAL TABLE REMOTE1_VT SET PROPERTY 'name1' = 'value1';
ALTER VIRTUAL TABLE REMOTE1_VT SET PROPERTY 'name1' = 'value2';
ALTER VIRTUAL TABLE REMOTE1_VT UNSET PROPERTY 'name1';
```

Set a new property for the virtual column A in virtual table REMOTE1_VT, then update and unset it.

```
ALTER VIRTUAL TABLE REMOTE1_VT ALTER A SET PROPERTY 'name1' = 'value1';
ALTER VIRTUAL TABLE REMOTE1_VT ALTER A SET PROPERTY 'name1' = 'value2';
ALTER VIRTUAL TABLE REMOTE1_VT ALTER A UNSET PROPERTY 'name1';
```

**Related Information**

- DROP TABLE Statement (Data Definition) [page 980]
- VIRTUAL_TABLES System View [page 1718]
- M_EXPENSIVE_STATEMENTS System View [page 1905]
4.10.1.27 ALTER WORKLOAD CLASS Statement (Workload Management)

Changes workload classes.

Syntax

```
ALTER WORKLOAD CLASS { <workload_class_name> { <inheritance> | <property_list> | ENABLE | DISABLE } | ALL { ENABLE | DISABLE } }
```

Syntax Elements

- **workload_class_name**
  Changes the specified workload class.
  ```
  <workload_class_name> ::= <identifier>
  ```

- **inheritance**
  Specifies a parent workload class.
  ```
  <inheritance> ::= PARENT { <parent_name> | NULL }
  ```

The workload class inherits the TOTAL STATEMENT MEMORY LIMIT or TOTAL STATEMENT THREAD LIMIT value from the parent workload class.

The child workload class must have the corresponding individual limit properties of the parent workload class and must not have aggregated properties, for example, a workload class with class-wise limits like total statement thread and memory limit.

The parent workload class must have a TOTAL STATEMENT MEMORY LIMIT or TOTAL STATEMENT THREAD LIMIT property and no individual properties, for example statement thread/memory limit.

Hierarchies can be only a single level (parent-child). Specifying NULL unsets the parent workload class.

- **property_list**
  Defines the properties of a workload class. The properties have the format of key-value pairs.
  ```
  property_list ::= [SET | UNSET] {<key> = '<value>'}[,<{'key'} = '<value>']} [...] 
  ```

  ```
  <key> ::= PRIORITY 
  STATEMENT MEMORY LIMIT 
  STATEMENT THREAD LIMIT 
  STATEMENT TIMEOUT 
  TOTAL STATEMENT MEMORY LIMIT 
  TOTAL STATEMENT THREAD LIMIT 
  ```
WRITE TRANSACTION LIFETIME
IDLE CURSOR LIFETIME
ADMISSION CONTROL REJECT CPU THRESHOLD
ADMISSION CONTROL REJECT MEMORY THRESHOLD
ADMISSION CONTROL QUEUE CPU THRESHOLD
ADMISSION CONTROL QUEUE MEMORY THRESHOLD

PRIORITY
Specifies a value from 0-9. A higher number specifies a higher priority.

STATEMENT MEMORY LIMIT
Specifies a limit in GB. For example, STATEMENT MEMORY LIMIT = '2' specifies a 2 GB limit.

STATEMENT THREAD LIMIT
Specifies a statement concurrency limit, rather than a thread limit; SAP HANA does not impose a thread limit on statements.

STATEMENT TIMEOUT
Specifies an expiry time for statement execution, after which statement execution is canceled and the transaction is rolled back. The default is 0 (no expiry).

TOTAL STATEMENT MEMORY LIMIT
Specifies an overall total memory limit, in GB, for all statements belonging to a specific workload class. If this property is specified, then PRIORITY and TOTAL STATEMENT THREAD LIMIT must also be specified.

TOTAL STATEMENT THREAD LIMIT
Specifies an overall limit for the number of active threads scheduled for statements for the specific workload class. If this property is specified, then PRIORITY and TOTAL STATEMENT MEMORY LIMIT must also be specified.

WRITE TRANSACTION LIFETIME
Specifies the duration of uncommitted write transactions, in minutes, before the connection is terminated. If this value is set to 0, then the timeout behavior is disabled.

IDLE CURSOR LIFETIME
Specifies the duration of cursors, in minutes, before the connection is terminated. If this value is set to 0, then the timeout behavior is disabled.

ADMISSION CONTROL REJECT CPU THRESHOLD / ADMISSION CONTROL REJECT MEMORY THRESHOLD
Specifies the threshold value as a percentage to reject a request based on CPU or memory consumption, if the measured load is equal to or larger than the configured value from the workload class. The threshold range is 0 - 100, where 0 rejects all requests and 100 accepts all requests. This property only applies to new incoming requests (statements already running are not affected).
A request rejection returns the message 'rejected by workload class configuration'.

Precedence is workload class, then session-wise admission control.

ADMISSION CONTROL QUEUE CPU THRESHOLD / ADMISSION CONTROL QUEUE MEMORY THRESHOLD
Specifies the threshold range as a value based on CPU or memory consumption. The threshold range is 0 - 100, where value 0 represents always queuing and value 100 represents no queuing. If the measured load is equal to or larger than configured queueing threshold from workload class, an arrived request is enqueued. If the waiting time in the queue of a certain request greater than the queue_timeout value, the issued request is rejected.
A request rejection returns the message 'queue wait timeout exceeded'.
This property only handles new incoming requests (already running statements are not affected) as with session-wise admission control.

Precedence is REJECTION comparison, then QUEUEING comparison. Workload class takes precedence over admission control.

**ENABLE or DISABLE**

Enables or disables the workload class.

**ALL** Changes all workload classes.

**Description**

A workload class with only NULL values behaves like the default workload class.

An aggregated workload class must have all properties specified (TOTAL STATEMENT MEMORY LIMIT, TOTAL STATEMENT THREAD LIMIT, PRIORITY).

**Precedence in the case of competing settings:** In the case of competing settings between workload class settings and user parameter settings (set by using ALTER USER), a workload class setting takes precedence over a user parameter setting. Workload class settings and user parameter settings always take precedence over .ini file settings. If the STATEMENT TIMEOUT setting, which can also be set by the client, is set to something more restrictive by the client than the value that is determined after evaluating the setting on the server side, then the more restrictive client-side setting is used instead. The following table demonstrates how precedence is determined:

<table>
<thead>
<tr>
<th>global.ini</th>
<th>User parameters</th>
<th>Workload class</th>
<th>Resulting effective values (statement execution)</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5</td>
<td>if not matched class found</td>
<td>priority 5, thread limit 5, memory limit 50GB</td>
</tr>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5, memory 5</td>
<td>priority 7, thread limit 10, memory limit 30GB</td>
<td>priority 7, thread limit 10, memory limit 30GB</td>
</tr>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5, memory 30GB</td>
<td>priority 7, thread limit 10, memory limit undefined</td>
<td>priority 7, thread limit 10, memory limit 30GB</td>
</tr>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5, memory 30GB</td>
<td>priority 7, total thread limit 10, total memory limit 30GB</td>
<td>priority 7, total thread limit 10, total memory limit 30GB</td>
</tr>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5, memory 30GB</td>
<td>priority 7, total thread limit undefined, total memory limit 50GB (SQL error returned at CREATE/ALTER WORKLOAD CLASS)</td>
<td>priority 5, thread limit 5, memory limit 30GB</td>
</tr>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5, memory 30GB</td>
<td>priority 7, total thread limit 10, total memory limit undefined (SQL error)</td>
<td>priority 5, thread limit 5, memory limit 30GB</td>
</tr>
</tbody>
</table>
### Examples

Unset the value for the STATEMENT_MEMORY_LIMIT for an existing workload class, MyWorkloadClass. The WORKLOAD_CLASSES view displays the value NULL for the column STATEMENT_MEMORY_LIMIT of this workload class. It also changes the PRIORITY to 5.

```sql
ALTER WORKLOAD CLASS "MyWorkloadClass"
  UNSET 'STATEMENT MEMORY LIMIT'
  SET 'PRIORITY' = '5';
```

Enable all workload classes and their related mappings.

```sql
ALTER WORKLOAD CLASS ALL ENABLE;
```

### Related Information

- CREATE WORKLOAD CLASS Statement (Workload Management) [page 929]
- WORKLOAD_CLASSES System View [page 1721]

### 4.10.1.28 ALTER WORKLOAD MAPPING Statement (Workload Management)

Changes workload mappings.

### Syntax

```sql
ALTER WORKLOAD MAPPING <mapping_name>
```
WORKLOAD CLASS <workloadclass_name> [ <property_list> [ <wildcard_option> ] ]

**Syntax Elements**

**mapping_name**

Specifies which workload mapping to change.

\[<mapping_name> ::= <identifier>\]

**workloadclass_name**

Specifies the workload class name.

\[<workloadclass_name> ::= <identifier>\]

**property_list**

Defines the workload mapping properties.

\[<property_list> ::= { [ SET <key_value_pair_list> ] | [ UNSET <key_value_pair_list> ] } \]
\[ <key_value_pair_list> ::= <key_value_pair> [, <key_value_pair> [,…] ] \]
\[ <key_value_pair> ::= '<key>'='<value>'\]

**<key> ::=**

APPLICATION USER NAME
| CLIENT
| APPLICATION COMPONENT NAME
| APPLICATION COMPONENT TYPE
| APPLICATION NAME
| OBJECT NAME
| XS APPLICATION USER NAME
| APPLICATION SOURCE

**<value> ::= <string_literal>**

USERGROUP NAME and USER NAME cannot be configured together. For established connections, changes to USERGROUP NAME are only applied when a connected database client reconnects. If there are two matched workload classes by USER NAME and USERGROUP NAME respectively, then USER NAME takes precedence over USERGROUP NAME.

SCHEMA NAME and OBJECT NAME must be configured together. SCHEMA NAME is the schema of the object. OBJECT NAME can be a procedure name, application function library (AFL) area or package name.

If an AFL area or package name is specified, then the workload class is applied to all AFFLLANG procedures that call AFs in that AFL area.

If there are multiple workload classes matched by SCHEMA NAME and OBJECT NAME, then the following order of precedence is used applied: AREA --> PACKAGE --> PROCEDURE.

XS APPLICATION USER NAME takes precedence over APPLICATION USER NAME.

APPLICATION SOURCE value must be set with setCommandInfo() method for it to be considered when mapping to a workload class.

**wildcard_option**
Specifies the wildcard character to use for value matching. All wildcard characters are interpreted as 0 or more characters.

```
<wildcard_option> ::= WITH WILDCARD [ <wildcard-character> ]
<wildcard-character> ::= <ASCII_character>
```

Wildcard characters can only be ASCII characters and cannot be used with USER NAME or USERGROUP NAME, SCHEMA NAME, or OBJECT NAME.

Only prefix wildcards are supported, for example WILDCARD %.

If no wildcard character is specified, then the default wildcard character is %. Use one wildcard character per property.

If you alter a workload mapping that already has a property with a wildcard and you don’t reset the wildcard option, then the property no longer has a wildcard associated with it.

**Permissions**

You need the WORKLOAD_ADMIN system privilege.

**Description**

Workload mapping settings are stored in the WORKLOAD_MAPPINGS system view.

**Examples**

Change the definition of the MyWorkloadMapping workload mapping to the ABCADM database user and unset the APPLICATION NAME application.

```
ALTER WORKLOAD MAPPING "MyWorkloadMapping" WORKLOAD CLASS "MyWorkloadClass"
    SET 'USER NAME' = 'ABCADM' UNSET 'APPLICATION NAME';
```

The following statement alters the MyWorkloadMapping1 workload mapping to use the wildcard character % by default for value matching:

```
ALTER WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyWorkloadClass" SET
    'APPLICATION NAME' = 'BW%' WITH WILDCARD;
```

The following statement alters the MyWorkloadMapping1 workload mapping to update the application name to be BW%* (without a wildcard):

```
ALTER WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyWorkloadClass" SET
    'APPLICATION NAME' = 'BW%*';
```
The following statement alters the MyWorkloadMapping1 workload mapping to set the user group name to USERGROUP_TEST.

```
ALTER WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyWorkloadClass" SET 'USERGROUP NAME' = 'USERGROUP_TEST';
```

The following statement alters an entry for the "BW_SYSTEM" workload mapping for the object SYS.CHECK_ES.

```
ALTER WORKLOAD MAPPING "BW_SYSTEM" WORKLOAD CLASS "DATAMART" SET 'SCHEMA NAME' = 'SYS', 'OBJECT NAME' = 'CHECK_ES';
```

The following statement alters an entry for the MyWorkloadMapping1 workload mapping for the XS application user name TESTER.

```
ALTER WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "DATAMART" SET 'XS APPLICATION USER NAME' = 'TESTER';
```

The following statement alters an entry for the MyWorkloadMapping1 workload mapping where APPLICATION SOURCE exactly matches example.cc:456.

```
ALTER WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyClass1" SET 'APPLICATION SOURCE' = 'example.cc:456';
```

The following statement alters the MyWorkloadMapping1 workload mapping to exclude APPLICATION SOURCE.

```
ALTER WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyClass1" UNSET 'APPLICATION SOURCE';
```

**Related Information**

- WORKLOAD_MAPPINGS System View [page 1723]
- CREATE WORKLOAD MAPPING Statement (Workload Management) [page 934]

### 4.10.1.29 ALTER X509 PROVIDER (Access Control)

Alters an X.509 provider in the SAP HANA database.

**Syntax**

```
ALTER X509 PROVIDER <x509_provider_name>
    { SET { [ <issuer_clause> ] [ <matching_rules_clause> ] } }
    | UNSET MATCHING RULES
```
Syntax Elements

**x509_provider_name**

Specifies the identifier of an X.509 provider to be modified.

```
<x509_provider_name> ::= <simple_identifier>
```

**issuer_clause**

Sets the issuer distinguished name for the X.509 provider.

```
<issuer_clause> ::= ISSUER <issuer_distinguished_name>
<issuer_distinguished_name> ::= <string_literal>
```

**matching_rules_clause**

Specifies one or more rules for matching external identities to database users. A matching rule is a distinguished name where one attribute has a value of '*'. This attribute contains the user name that needs to be matched during logon. All other attributes must match and be in the same order.

Matching rules are tried in the same order they are defined in the provider, and the first one that matches is used.

```
<matching_rules_clause> ::= MATCHING RULES
  '<subject_distinguished_name_mapping>' [, '<subject_distinguished_name_mapping>' ... ]
  <subject_distinguished_name_mapping> ::= <string_literal>
```

Setting new matching rules replaces existing rules for the provider.

**UNSET MATCHING RULES** Removes matching rules for the provider. If a user is mapped to the provider using a wildcard (ANY) subject, you cannot unset the matching rules.

Description

For the SET clause, at least one of the two optional parameters must be specified.

Permissions

Only database users with the USER ADMIN system privilege are allowed to change X.509 providers.

Examples

Change the issuer of the MyProvider X.509 provider.

```
ALTER X509 PROVIDER MyProvider SET ISSUER 'CN=DigiCert Global Root CA,
OU=www.digicert.com, O=DigiCert Inc, C=US'
```
Remove the matching rules of the MyProvider X.509 provider.

```
ALTER X509 PROVIDER MyProvider UNSET MATCHING RULES
```

**Related Information**

- X.509 Certificate-Based User Authentication
- CREATE X509 PROVIDER (Access Control) [page 937]
- DROP X509 PROVIDER (Access Control) [page 991]
- X509_PROVIDERS System View [page 1725]
- X509_PROVIDER_RULES System View [page 1725]
- X509_USER_MAPPINGS System View [page 1726]

### 4.10.1.30 ANNOTATE Statement (Data Definition)

Annotates SQL objects such as tables, views, columns, table functions, procedures, and parameters.

**Syntax**

```
ANNOTATE { [ <table_annotations> ] | [ <column_annotations> ] | [ <parameter_annotations> ] )  
<table_annotations> ::= { [ <key_set_operation> ] [ <key_unset_operation> ] }  
<column_annotations> ::= COLUMN <column_ref> { <key_set_operation> | <key_unset_operation> }  
<parameter_annotations> ::= <parameter_annotation> [ <parameter_annotation> [...] ]  
<parameter_annotation> ::= PARAMETER <column_ref> { <key_set_operation> | <key_unset_operation> }  
<key_set_operation> ::= SET <key_value_pair_list>  
<key_unset_operation> ::= UNSET <key_list>  
=key_value_pair_list> ::= <key_value_pair> [, <key_value_pair> [,...] ]  
<key_value_pair> ::= '<key>='<value>'  
<key_list> ::= { '<key>'[,'<key>'[,...] ] | ALL }  
```

**Syntax Elements**

- **table_ref**
  Specifies a table, view, table function, or procedure.

```sql
<table_ref> ::= [ <schema> ).{ <table_name> | <view_name> }  
```

- **COLUMN column_ref**
Specifies a column of a table or view.

\[
\text{<column_ref>} ::= \text{COLUMN [ <schema>.] (<table_name> | <view_name>).<column_name>}
\]

**PARAMETER parameter_ref**

Specifies a parameter of a procedure or function.

\[
\text{<parameter_ref>} ::= \text{PARAMETER [ <schema>.] (<procedure_name> | <function_name>).<parameter_name>}
\]

**annotation_operation**

Specifies the annotation to perform.

\[
\text{<annotation_operation>} ::= \\
\text{SET <key> = <value>} \\
\text{UNSET [ <key> | ALL ]}
\]

Use SET to annotate the specified database object with a key/value pair. If the key already exists for the database object, then the existing value is replace by <value>.

Use UNSET to remove the key/value pair identified by <key> for the specified database object, or use UNSET ALL to remove all key/value pair annotations for the specified database object.

<key> and <value> are string literals that specify the annotation as a key/value pair (for example, 'key1=20')

**Description**

SQL Annotation, a form of metadata, provides additional insight into an SQL object.

Annotated values for database objects are stored in the ANNOTATIONS system view.

**Permissions**

You must have ALTER privileges on the object to annotate it.

**Examples**

The following statements create a function and annotate it:

```sql
CREATE FUNCTION add_shipping_func(price DECIMAL(15,2), shipping DECIMAL(15,2))
RETURNS result DECIMAL(15,2)
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER AS
BEGIN
result := :price + :shipping;
END;
```
The following statements replaces the value for the `currency` annotations on the `price` parameter:

```
ANNOTATE PARAMETER add_shipping_func.price SET 'currency'='CAD';
```

The following statement removes the `currency` annotation on the `shipping` parameter:

```
ANNOTATE PARAMETER add_shipping_func.shipping UNSET 'currency';
```

The following statement creates a table, adds some annotations, and then removes them all.

```
CREATE COLUMN TABLE t1 ( col1 INT NOT NULL);
ANNOTATE t1 SET 'priority'='high';
ANNOTATE COLUMN t1.col1 SET 'datatype'='integer';
ANNOTATE COLUMN t1.col1 SET 'nullability'='no nulls';
ANNOTATE COLUMN t1.col1 UNSET ALL;
ANNOTATE t1 UNSET 'priority';
```

### Related Information

ANNOTATIONS System View [page 1494]

---

**4.10.1.31 BACKUP CANCEL Statement (Backup and Recovery)**

Cancels a running data backup.

#### Syntax

```
BACKUP CANCEL [FOR <database_name>] <backup_ID>
```

#### Syntax Elements

- **database_name**

  Specifies the name of an SAP HANA database.

  ```
  <database_name> ::= <identifier>
  ```

- **backup_ID**
Specifies the backup to be canceled.

<backup_ID> ::= <integer>

**Description**

Requirements:

- Executed on the system database only
- The system and the tenant database(s) must be online

1. Find the backup ID.
   When a data backup is started, the system assigns a unique ID to the data backup. To find the backup ID of the running data backup, use the monitoring view M_BACKUP_CATALOG, which provides an overview of information about backup and recovery activities.
   Use the following SQL statement to display the backup ID:

   ```sql
   SELECT BACKUP_ID FROM "M_BACKUP_CATALOG"    WHERE ENTRY_TYPE_NAME = 'complete data backup'    AND STATE_NAME = 'running'    ORDER BY SYS_START_TIME DESC;
   ```

2. To cancel the running data backup, use the following SQL statement:

   ```sql
   BACKUP CANCEL <backup_ID>
   ```

   The data backup is canceled asynchronously.

3. Check that the data backup was canceled.
   Use the monitoring view M_BACKUP_CATALOG to display the status of the canceled data backup. The status canceled confirms that the backup was canceled successfully. When the backup status is canceled, you can start a new data backup.

If you cancel a running data backup that is waiting for other resources, the cancelation is postponed until the cause of the wait situation has been resolved. Until that time, the backup is flagged as cancel pending.

**Permissions**

Privileges needed:

- BACKUP OPERATOR or BACKUP ADMIN
  FOR <database_name>: DATABASE BACKUP OPERATOR, DATABASE BACKUP ADMIN, or DATABASE ADMIN

**Example**

Cancel the data backup with ID **1331715084250**:

```sql
BACKUP CANCEL 1331715084250;
```
4.10.1.32 BACKUP CATALOG DELETE Statement (Backup and Recovery)

Deletes the records of all backups (both in the file system and through Backint) from the backup catalog. Optionally, also deletes the associated physical backups.

Syntax

```
BACKUP CATALOG DELETE [ FOR <database_name> ]      
| { ALL BEFORE BACKUP_ID <backup_id> [ WITH FILE | WITH BACKINT | COMPLETE ]       | BACKUP_ID <backup_id> [ COMPLETE ] }
```

Syntax Elements

FOR database_name

Specifies the name of the tenant database whose backups will be deleted.

```
<database_name> ::= <identifier>
```

FOR <database_name> can only be used through the system database.

If FOR <database_name> is omitted, then the local database is assumed. The local database can be either the system database or a tenant database.

backup_id

The backup ID uniquely identifies the backup to be deleted.

ALL BEFORE <backup_id> removes the backup catalog records of backups that are older than the specified backup.

WITH FILE

In addition to deleting the records in the backup catalog (for both file-based and Backint backups), WITH FILE also deletes the associated physical backups in the file system.

Note

SAP HANA can only physically delete file-based backups that are at the location specified in the backup catalog.

WITH BACKINT

In addition to deleting the records in the backup catalog (for both file-based and Backint backups), WITH BACKINT also deletes the associated physical backups in the third-party backup tool.

COMPLETE

In addition to deleting the records in the backup catalog (for both file-based and Backint backups), COMPLETE also deletes the associated physical backups in the file system and the third-party backup tool.
SAP HANA can only delete file-based physical backups that are at the location specified in the backup catalog.

Description

BACKUP CATALOG DELETE can be executed on the system database or on a tenant database.

From the system database, you can execute BACKUP CATALOG DELETE on a tenant database, which is specified using the FOR <database_name> clause. If a tenant database is specified, that database must be online.

Tip

It is recommended to regularly delete backups that are no longer needed.

It is important to regularly truncate the backup catalog because, as the backup catalog increases in size, it takes longer to record each new backup.

What Backups are Deleted?

If you omit COMPLETE, WITH FILE, or WITH BACKINT, then only the specified records in the backup catalog are deleted. The physical backups are not deleted.

The ALL BEFORE option deletes the log backups and the log backup records in the backup catalog only.

At least one data backup must remain in the backup catalog. For this reason, if a selected data backup is the only remaining backup, that backup cannot be deleted.

Note

The records in the backup catalog are deleted synchronously. Physical backups are deleted asynchronously.

This means that the deletion process can continue, even after a service shutdown or a restart.

Backup Retention

Complete data backups that are flagged as RETAINED cannot be deleted using BACKUP CATALOG DELETE.

For more information, see What Information is in the Backup Catalog? in SAP HANA Administration Guide for SAP HANA Platform.

Backup Catalog and Recovery

If a data backup is physically available, but is not recorded in the backup catalog, that data backup can still be used to recover the database. Without a backup catalog, a point-in-time recovery cannot be done. Without a backup catalog, it is only possible to recover SAP HANA to a specific complete data backup.

Third-party data backups that are not recorded in the backup catalog must have a unique prefix to be able to be used for a recovery.
For more information, see *Housekeeping: Deleting and Archiving Backups* in *SAP HANA Administration with SAP HANA Cockpit (Backup and Recovery)*.

**Permissions**

Privileges needed:

```
BACKUP ADMIN
FOR <database_name>: DATABASE
```

**Examples**

**Example 1:** Truncate the backup catalog and remove all the physical backups older than the data backup specified by BACKUP_ID `1496915612668`. The statement deletes the associated data backups, delta backups, log backups, and backups of the backup catalog in both the file system and in the third-party backup tool.

```
BACKUP CATALOG DELETE ALL BEFORE BACKUP_ID 1496915612668 COMPLETE;
```

This statement can be executed from the system database or the tenant database.

**Example 2:** For the tenant database `PR1`, the following example truncates the backup catalog and removes all the physical backups older than the data backup specified by BACKUP_ID `1496915612668`. It deletes the associated data backups, delta backups, log backups, and backups of the backup catalog in both the file system and in the third-party backup tool.

```
BACKUP CATALOG DELETE FOR PR1 ALL BEFORE BACKUP_ID 1496915612668 COMPLETE;
```

This statement can be executed from the system database.

**Example 3:** The following example only deletes the record of the data backup specified by BACKUP_ID `1496915612668` in the backup catalog. The physical data backup itself is not deleted.

```
BACKUP CATALOG DELETE BACKUP_ID 1496915612668;
```

This statement can be executed from the system database or the tenant database.

**Related Information**

*Housekeeping: Deleting and Archiving Backups (SAP HANA Administration with SAP HANA Cockpit)*

*What Information is in the Backup Catalog? (SAP HANA Administration Guide for SAP HANA Platform)*
4.10.1.33 BACKUP CATALOG RETAINED Statement (Backup and Recovery)

Flag or unflag complete data backups as RETAINED to prohibit or allow their deletion.

Syntax

BACKUP CATALOG [ SET | UNSET ] RETAINED [ FOR <database_name> ] BACKUP_ID <backup_id>

Syntax Elements

FOR database_name

Specifies the name of the database.

FOR <database_name> can only be used through the system database.

If FOR <database_name> is omitted, the local database is assumed. The local database can be either the system database or a tenant database.

backup_id

The backup ID uniquely identifies the complete data backup.

Description

Successfully saved complete data backups can be flagged as RETAINED when they are created. After a data backup has been created, you can use BACKUP CATALOG [ SET | UNSET ] RETAINED to remove or add the RETAINED flag.

Complete data backups that are flagged as RETAINED cannot be deleted by SAP HANA, either individually, by scheduled housekeeping tasks, or by the SQL statement BACKUP CATALOG DELETE.

For more information, see BACKUP CATALOG DELETE Statement (Backup and Recovery) and What Information is in the Backup Catalog? in SAP HANA Administration Guide for SAP HANA Platform.

BACKUP CATALOG [ SET | UNSET ] RETAINED can be executed on the system database or on a tenant database.

From the system database, you can execute BACKUP CATALOG [ SET | UNSET ] RETAINED on a tenant database, which is specified using the FOR <database_name> clause. If a tenant database is specified, that database must be online.

To see an overview of the data backups that are flagged as RETAINED, use the system view M_BACKUP_CATALOG.
Without a RETAINED flag, a data backup can be deleted if the appropriate prerequisites are met.

**Note**

SAP HANA will not delete a data backup if it is the only data backup remaining in a backup catalog.

For more information, see *Prerequisites for Deleting Old Backups* in *SAP HANA Administration with SAP HANA Cockpit*.

**Permissions**

Privileges needed:

BACKUP ADMIN

FOR <database_name>: DATABASE BACKUP ADMIN OR DATABASE ADMIN

**Examples**

**Example 1:** Flag a complete data backup with ID 13546473 as RETAINED. This backup can now not be deleted by SAP HANA, either individually, by scheduled housekeeping tasks, or by the SQL statement BACKUP CATALOG DELETE.

```
BACKUP CATALOG SET RETAINED BACKUP_ID 13546473;
```

**Example 2:** Remove the RETAINED flag from a complete data backup with ID 13546473. SAP HANA will now not prohibit requests to delete this backup.

```
BACKUP CATALOG UNSET RETAINED BACKUP_ID 13546473;
```

**Related Information**

BACKUP CATALOG DELETE Statement (Backup and Recovery) [page 690]
M_BACKUP_CATALOG System View [page 1748]
What Information is in the Backup Catalog? (SAP HANA Administration Guide for SAP HANA Platform)
Prerequisites for Deleting Old Backups (SAP HANA Administration with SAP HANA Cockpit)
4.10.1.34 BACKUP CHECK Statement (Backup and Recovery)

Checks that required disk space for a backup is available in a specified location.

Syntax

```
BACKUP CHECK [FOR <database_name>] <data_file_definition> SIZE <size>
```

Syntax Elements

**database_name**

Specifies the database to be checked.

```
<database_name> ::= <identifier>
```

The database name can only be specified for an SAP HANA multitenant system.

**data_file_definition**

Specifies the location in the file system to be checked for sufficient space.

```
<data_file_definition> ::= USING FILE ( <file_name_prefix> )
<file_name_prefix> ::= <string_const>
```

**size**

Specifies the amount of memory in bytes that is required for the backup.

```
<size> ::= <integer>
```

Description

If insufficient space is available or the folder does not exist, then an error message is returned.

Requirements:

- Executed on the system database only
- The system database must be online
- If a `<database_name>` is specified, then the specified database must be online as well

Permissions

Privileges needed:
BACKUP OPERATOR or BACKUP ADMIN
FOR <database_name>: DATABASE BACKUP OPERATOR, DATABASE BACKUP ADMIN, or DATABASE ADMIN

Example

Check whether 50GB of space are available in the /backup/data folder:

```
BACKUP CHECK USING FILE ('/backup/data') SIZE 53687091200;
```

4.10.1.35 BACKUP CHECK ACCESS Statement (Backup and Recovery)

Checks whether a backup can be accessed, and the integrity of its metadata.

Syntax

```
BACKUP CHECK ACCESS [FOR <database>] BACKUP_ID <int_const>     
[USING SOURCE '<system_id>'] [USING CATALOG PATH (<path>)]
```

Syntax Elements

for_database

Checks if a specified backup can be accessed by the tenant database.

```
<for_database> ::= FOR <database_name>
```

This command is only available for SAP HANA tenant databases.

int_const

Specifies the BACKUP ID of the full data backup to use for recovery.

For example, BACKUP_ID 1540499452637.

system_source_definition

```
<system_source_definition> ::= USING SOURCE '<system_id>'
```

path_definition

```
<path_definition> ::= [USING CATALOG PATH ( <path> )]
```
In the file system, SAP HANA searches the specified locations for the backup catalog in the following order:

<table>
<thead>
<tr>
<th>Path Specified...</th>
<th>Where Searched...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the SQL statement (optional):</td>
<td>The path specified is searched for a backup catalog.</td>
</tr>
<tr>
<td>USING CATALOG PATH</td>
<td>No subdirectories are searched.</td>
</tr>
<tr>
<td>2. Configured path.</td>
<td>The path to be searched is specified by the following parameter:</td>
</tr>
<tr>
<td></td>
<td>basepath_catalogbackup</td>
</tr>
<tr>
<td></td>
<td>This parameter is in the persistence section of the global.ini parameter file.</td>
</tr>
</tbody>
</table>

In the file system, SAP HANA searches the specified locations for backups in the following order:

<table>
<thead>
<tr>
<th>Path Specified...</th>
<th>Where Searched...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the SQL statement (optional):</td>
<td>The specified paths are searched for data backups or log backups.</td>
</tr>
<tr>
<td>USING LOG PATH &lt;path list&gt;</td>
<td>All subdirectories are searched.</td>
</tr>
<tr>
<td>USING DATA PATH &lt;path&gt;</td>
<td>If a specified path is relative, the path is based on the backup paths specified by the following parameters:</td>
</tr>
<tr>
<td></td>
<td>basepath_databackup</td>
</tr>
<tr>
<td></td>
<td>basepath_logbackup</td>
</tr>
<tr>
<td></td>
<td>These parameters are in the persistence section of the global.ini parameter file.</td>
</tr>
<tr>
<td>2. Configured paths.</td>
<td>The paths to be searched are specified by the following parameters:</td>
</tr>
<tr>
<td></td>
<td>basepath_databackup</td>
</tr>
<tr>
<td></td>
<td>basepath_logbackup</td>
</tr>
<tr>
<td></td>
<td>These parameters are in the persistence section of the global.ini parameter file.</td>
</tr>
<tr>
<td>3. If no backups are found in the paths for 1. or 2.</td>
<td>The paths specified in the original backup catalog are searched for the backups.</td>
</tr>
</tbody>
</table>
**Description**

This statement checks whether a backup can be accessed, and the integrity of its metadata.

`BACKUP CHECK ACCESS` only checks an individual backup. Conversely, the `CHECK ACCESS` option for `RECOVER DATABASE` checks whether all the backups required for a recovery are available.

`hdbbackupdiag` `BACKUP CHECK ACCESS` performs the same function as the `--check` option of `hdbbackupdiag`. However, with `BACKUP CHECK ACCESS`, the database must be offline. With `hdbbackupdiag`, the database can be either offline or online.

`BACKUP CHECK ACCESS` only checks for a single complete data backup identified by a specific backup ID. `BACKUP CHECK ACCESS` does not check for log backups or delta backups.

`hdbbackupdiag --check` checks whether all the backups (including log backups and delta backups) required for a recovery exist and can be accessed, and also that the metadata is consistent.

Unlike `hdbbackupdiag --check`, the `hdbbackupcheck` tool checks the content of the backups on page level using checksums.

For more information about `hdbbackupcheck`, see SAP Note 1869119 (Checking backups with "hdbbackupcheck").

**Error Handling**

The result set of `BACKUP CHECK ACCESS` is written to an XML file named `strategyOutput.xml` in the SAP HANA trace directory.

If an error occurs, the `strategyOutput.xml` file is deleted.

An SQL error is not always returned. If `strategyOutput.xml` was not created or was rewritten, check the `backup.log` for error messages.

For more information, see `backup.log` in the SAP HANA Administration Guide (SAP HANA Database Backup and Recovery).

**Permissions**

Privilege needed:

`FOR <database_name>: DATABASE RECOVERY OPERATOR or DATABASE ADMIN`

**Related Information**

SAP Note 1869119
backup.log (SAP HANA Administration Guide (SAP HANA Database Backup and Recovery))
Change the Default Destination for File-Based Data Backups
Change the Log Backup Destination
4.10.1.36 BACKUP DATA Statement (Backup and Recovery)

Creates a data backup of the specified database in the specified location.

Syntax

```
BACKUP DATA [ <backup_delta> ] [ FOR <database_name> ]
<data_backup_definition> [ <additional_option_list> ] [ <comment> ]
```

Syntax Elements

**backup_delta**
Specifies the type of backup to be created.

```
<backup_delta> ::= INCREMENTAL | DIFFERENTIAL
```

If this parameter is omitted, a complete data backup is created.

**database_name**
If you are connected to the system database, specifies the name of the tenant database to back up.

```
<database_name> ::= <identifier>
```

**data_backup_definition**
Defines the file location where the backup is created.

```
<data_backup_definition> ::= USING { <data_backup_definition_file> | <data_backup_definition_backint>
[DISABLE ES ENCRYPTION CHANGE] }
<data_backup_definition_file> ::= FILE ( '<prefix>' | '<path>', '<prefix>' )
<data_backup_definition_backint> ::= BACKINT ( '<prefix>' | '<path>', '<prefix>' )
```

You can provide the path and prefix either as a single string or as two separate strings for path and prefix. The trailing path delimiter in <path> may be omitted.

**DISABLE ES ENCRYPTION CHANGE**
Applies to SAP HANA dynamic tiering only.

When you specify the DISABLE ES ENCRYPTION CHANGE clause, extended storage copies blocks from the database to the backup archive without modifying their encryption state. This can improve backup and recovery speeds for extended storage.

Changes to the encryption state during recovery are not permitted with speeds for extended storage.

SAP recommends you do not specify DISABLE ES ENCRYPTION CHANGE when data volume encryption is enabled.
If backup encryption is enabled, specifying `DISABLE ES ENCRYPTION CHANGE` results in an error.

You can monitor the `DISABLE ES ENCRYPTION CHANGE` allowed or disallowed status in the `M_BACKUP_CATALOG` system view.

**additional_option_list**

Additional options for backup.

```plaintext
<additional_option_list> ::= <additional_option> [...
<additional_option> ::= ASYNCHRONOUS | TOOLOPTION '<parameter_string>' | COMPRESSED | RETAINED | INCLUDE CONFIGURATION | EXCLUDE CONFIGURATION
```

**ASYNCHRONOUS**

With `ASYNCHRONOUS`, the system does not wait for the backup process to end. The status of the backup is visible in the `M_BACKUP_CATALOG` system view. By default, a synchronous backup is created.

**TOOLOPTION**

`TOOLOPTION` is only available for `BACKINT`.

`TOOLOPTION` passes a vendor-specific parameter string to a third-party backup tool. The string has a maximum length of 4096 bytes.

**COMPRESSED**

`COMPRESSED` creates a data backup or delta backup (differential or incremental backup) using native SAP HANA compression. Currently, SAP HANA supports the LZ4 compression algorithm only.

The compression level is defined using the parameter `data_backup_compression_level`. For more information, see Configure Data Backup Compression in the SAP HANA Administration Guide for SAP HANA Platform (SAP HANA Database Backup and Recovery).

**RETAINED**

You can flag complete data backups as `RETAINED`.

Backups flagged as `RETAINED` cannot be deleted by SAP HANA, either individually, by scheduled housekeeping tasks, or by the SQL statement `BACKUP CATALOG DELETE`.

For more information, see What Information is in the Backup Catalog? in SAP HANA Administration Guide for SAP HANA Platform.

To change the `RETAINED` status for complete data backups, use the SQL statement `BACKUP CATALOG RETAINED`.

For more information, see `BACKUP CATALOG RETAINED` Statement (Backup and Recovery).

**INCLUDE CONFIGURATION | EXCLUDE CONFIGURATION**

`INCLUDE CONFIGURATION` includes the user configuration files (changed `.ini` files) in a data backup.

The user configuration files are stored without encryption in the metadata of the data backup. The user configuration files are not compressed.

Data backups of tenant databases include the user configuration files in the `DATABASE` layer.
Data backups of the system database include the user configuration files in the SYSTEM layer. This also includes the nameserver.ini. However, nameserver.ini files cannot be restored as part of a recovery. You can manually restore the backup of the nameserver.ini file. To display the content of the backed-up nameserver.ini files, you can use hdbbackupcheck.

For more information, see Check Individual Backups Inside an SAP HANA Installation (hdbbackupdiag, hdbbackupcheck).

EXCLUDE CONFIGURATION creates a data backup without user configuration files.

**i Note**

The **INCLUDE CONFIGURATION | EXCLUDE CONFIGURATION** option with the SQL statement **BACKUP DATA** overrides the setting for **include_configuration_backup**.

For more information, see Include User Configuration in Backup or Recovery in the SAP HANA Administration Guide for SAP HANA Platform (SAP HANA Database Backup and Recovery).

**comment**

Specifies a descriptive comment for the backup.

```sql
<comment> ::= COMMENT '<string_const>'
```

**Description**

Creates a data backup of the specified database in the specified location.

**i Note**

If a data backup with the same name already exists in the specified location, it will be overwritten by the newest data backup.

To set up data backup in batch mode, set up a batch user and authorize this user to create backups.

For batch processing with SAP HANA HDBSQL, use the SAP HANA secure user store (hdbuserstore). Otherwise, the password used to connect to the database is visible in the process list. hdbuserstore is installed as part of the client package.

For data backups for dynamic tiering extended storage, the global system property `es_data_backup_buffer_size` is a tuning option that may optimize backup performance.

**Permissions**

Privileges needed:

**BACKUP OPERATOR** or **BACKUP ADMIN**

**FOR <database_name>:** DATABASE BACKUP OPERATOR, DATABASE BACKUP ADMIN, or DATABASE ADMIN
**Examples**

1. Create a data backup of the entire database. This creates a data backup file of every relevant service in the default backup destination. Label the backup 'data backup of 2018-10-16' by using the `COMMENT` clause.

   ```sql
   BACKUP DATA USING FILE ('COMPLETE') COMMENT 'data backup of 2018-10-16';
   ```

   The data backup files are prepended with the prefix COMPLETE. The comment string is recorded in the backup catalog and is visible in the `M_BACKUP_CATALOG` system view. The following files are created in the default data backup directory:
   - `COMPLETE_databackup_0_1`
   - `COMPLETE_databackup_1_1`
   - `COMPLETE_databackup_2_1`

2. Create a data backup of the entire database and create a data backup for every relevant service in the default backup destination.

   ```sql
   BACKUP DATA USING FILE ('COMPLETE');
   ```

   The data backup files are prepended with the prefix COMPLETE. The following files are created in the default data backup directory:
   - `COMPLETE_databackup_0_1`
   - `COMPLETE_databackup_1_1`
   - `COMPLETE_databackup_2_1`

3. Create a data backup of the entire database and create a data backup for every relevant service in the directory `/backup` and the prefix `MONDAY`.

   ```sql
   BACKUP DATA USING FILE ('/backup/MONDAY');
   ```

   Create the same file using alternative syntax with separate strings to specify path and prefix.

   ```sql
   BACKUP DATA USING FILE ('/backup/', 'MONDAY');
   ```

   Create the same file using alternative syntax with separate strings to specify path and prefix, omitting the last path delimiter.

   ```sql
   BACKUP DATA USING FILE ('/backup', 'MONDAY');
   ```

   Data backup files are prepended with the string MONDAY. The following files are created:
   - `/backup/MONDAY_databackup_0_1`
   - `/backup/MONDAY_databackup_1_1`
   - `/backup/MONDAY_databackup_2_1`

4. Create a data backup of the entire database. This creates a data backup file for every relevant service in the directory `/backup`.

   ```sql
   BACKUP DATA USING FILE ('/backup/') ASYNCHRONOUS;
   ```

   Data backups are not labeled. The `ASYNCHRONOUS` option returns to the client immediately after starting the backup operation, even though the data backup is still running. The following backups are created:
   - `/backup/databackup_0_1`
   - `/backup/databackup_1_1`
5. Create a data backup of the entire instance. This creates a data backup for each service using the third-party backup tool.

```
BACKUP DATA USING BACKINT ('MyBackIntBackup') TOOLOPTION 'This is my tool option';
```

Data backups are prepended with the prefix `MyBackIntBackup`. The TOOLOPTION string `'This is my tool option'` is forwarded to the third-party backup tool. The following backups are created in the third-party backup tool:

- `MyBackIntBackup_databackup_0_1`
- `MyBackIntBackup_databackup_1_1`
- `MyBackIntBackup_databackup_2_1`

6. Create an incremental backup in the default data backup directory. Use the prefix `2018-10-16`.

```
BACKUP DATA INCREMENTAL USING FILE ('2018-10-16');
```

The example SQL statement would create backups with the following names (all on one line):

- `2018-10-16_databackup_incremental_1426237023821_1426237028496_0_1`
- `2018-10-16_databackup_incremental_1426237023821_1426237028496_1_1`
- `2018-10-16_databackup_incremental_1426237023821_1426237028496_2_1`
- `2018-10-16_databackup_incremental_1426237023821_1426237028496_3_1`

7. Create a differential backup based on the last data backup.

```
BACKUP DATA DIFFERENTIAL USING FILE ('2018-10-16');
```

The example SQL statement would create backups with the following names (all on one line):

- `2018-10-16_databackup_differential_1426237023821_1426237780534_0_1`
- `2018-10-16_databackup_differential_1426237023821_1426237780534_1_1`
- `2018-10-16_databackup_differential_1426237023821_1426237780534_2_1`
- `2018-10-16_databackup_differential_1426237023821_1426237780534_3_1`

8. (SAP HANA dynamic tiering only.)
Create a data backup of the entire database. Extended storage copies blocks from the database to the backup archive without modifying their encryption state.

```
BACKUP DATA USING FILE ('my_backup') DISABLE ES ENCRYPTION CHANGE;
```

Changes to the encryption state during recovery are not permitted with these backups. SAP recommends you do not specify `DISABLE ES ENCRYPTION CHANGE` when data volume encryption is enabled.

9. Create a compressed data backup of a tenant database called `MY_TENANT` from a third-party backup tool.

```
BACKUP DATA FOR MY_TENANT USING BACKINT ('2021-03-03-compressed') COMPRESSED COMMENT 'Compressed data backup from external storage';
```

**Related Information**

- `M_BACKUP_CATALOG System View` [page 1748]
4.10.1.37 BACKUP DATA CLOSE SNAPSHOT Statement (Backup and Recovery)

Confirms that a data snapshot has been created or that the data snapshot has been abandoned.

**Syntax**

```sql
BACKUP DATA FOR FULL SYSTEM CLOSE SNAPSHOT BACKUP_ID <backup_id>    { SUCCESSFUL <external_id> | UNSUCCESSFUL [ <string> ] }
```

**Syntax Elements**

- **FOR FULL SYSTEM**
  Closes a data snapshot of the system database and all the tenant databases.

- **FOR FULL SYSTEM** is mandatory.

- **BACKUP_ID backup_id**
  Specifies the unique ID of a data snapshot to be confirmed or abandoned.

- **SUCCESSFUL external_id**
  Specifies an ID for the data snapshot to help to identify a specific snapshot.

- **TIMESTAMP** is supported for `<external_id>`. The maximum length of the external ID is 64 characters.

- **UNSUCCESSFUL [string]**
  UNSUCCESSFUL `<string>` flags the data snapshot as unsuccessful in the backup catalog, and includes the comment specified in `<string>`. UNSUCCESSFUL without `<string>` flags the data snapshot as unsuccessful in the backup catalog without a comment.

  The maximum length of `<string>` is 256 characters.
## Description

To close a data snapshot:

<table>
<thead>
<tr>
<th>To Confirm or Abandon...</th>
<th>Perform the Following Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm the data snapshot</td>
<td>A data snapshot can be used to recover a database provided that it was completed successfully. If you wish to retain the data snapshot to use for a recovery later, you must first manually copy the data snapshot (the content of the data area) to a separate storage location. Then, you need to close (confirm) the data snapshot. Closing the data snapshot confirms that a data snapshot has been created and stored in a separate storage location, or that the data snapshot has been abandoned. When the data snapshot has been confirmed or abandoned, it is deleted from the data area. After that, it is possible to create further data snapshots or full backups. To identify the data snapshot during a subsequent recovery, you can specify an external ID. To find the backup ID, search in the monitoring view M_BACKUP_CATALOG. The comment that was used when creating the backup is displayed in the <code>COMMENT</code> column of M_BACKUP_CATALOG. If BACKUP DATA CLOSE SNAPSHOT returns an error, the data snapshot cannot be used. If an error occurs executing BACKUP DATA CLOSE SNAPSHOT, use BACKUP DATA DROP SNAPSHOT to remove the latest data snapshot completely from persistence.</td>
</tr>
<tr>
<td>Abandon the data snapshot</td>
<td>If you do not wish to retain the data snapshot, you only need to close (abandon) it.</td>
</tr>
</tbody>
</table>

After you have confirmed or abandoned a data snapshot, it is recorded in the backup catalog as either successful or unsuccessful.

## Permissions

Privileges needed:

BACKUP OPERATOR or BACKUP ADMIN
Examples

Example 1: Confirm that a data snapshot has been successfully created.

```
BACKUP DATA FOR FULL SYSTEM CLOSE SNAPSHOT BACKUP_ID 1400761459486 SUCCESSFUL 'External_ID_23';
```

Example 2: Abandon a data snapshot.

If the data snapshot was not created successfully, or if you do not want to use it, you can add a comment to help identify the data snapshot in the backup catalog. It can be helpful to add the reason for abandoning the snapshot.

```
BACKUP DATA FOR FULL SYSTEM CLOSE SNAPSHOT BACKUP_ID 1400761459486 UNSUCCESSFUL 'Do not use - was manually terminated';
```

Related Information

- BACKUP DATA DROP SNAPSHOT Statement (Backup and Recovery) [page 708]
- BACKUP DATA CREATE SNAPSHOT Statement (Backup and Recovery) [page 706]

4.10.1.38 BACKUP DATA CREATE SNAPSHOT Statement (Backup and Recovery)

Creates an internal database snapshot for a database instance.

**Syntax**

```
BACKUP DATA FOR FULL SYSTEM CREATE SNAPSHOT [ COMMENT <string> ] [ TIMEOUT <integer> ]
```

**Syntax elements**

**FOR FULL SYSTEM**

Creates an internal database snapshot of the system database and all the tenant databases.

**FOR FULL SYSTEM** is mandatory.

**COMMENT string**

Specifies a comment that is added for the data snapshot.
The maximum comment length is 256 characters and the comment appears in the COMMENT column of the monitoring view M_BACKUP_CATALOG.

TIMEOUT integer

Specifies a timeout (in seconds) for the operation.

Description

Creates an internal database snapshot that reflects a consistent database state at the point in time when the snapshot was created.

To use a data snapshot for a subsequent recovery, you then need to copy the data area of the system database and all the tenant database to a separate storage location, then close (confirm) the data snapshot.

If an internal database snapshot exists, no new data backups or new database snapshots can be created. Conversely, while a data backup is running, you cannot create a database snapshot.

Permissions

Privileges needed:

BACKUP OPERATOR or BACKUP ADMIN

Example

Prepare the database for a data snapshot by creating an internal database snapshot.

The comment MySnapshot helps to identify the data database snapshot in the backup catalog.

```
BACKUP DATA FOR FULL SYSTEM CREATE SNAPSHOT COMMENT 'MySnapshot';
```

Related Information

BACKUP DATA CLOSE SNAPSHOT Statement (Backup and Recovery) [page 704]
4.10.1.39 BACKUP DATA DROP SNAPSHOT Statement
(Backup and Recovery)

Physically deletes a data snapshot.

Syntax

```
BACKUP DATA FOR FULL SYSTEM DROP SNAPSHOT
```

Syntax elements

```
FOR FULL SYSTEM
```

Specifies that the data snapshot of the system database and all the tenant databases is deleted.

FOR FULL SYSTEM is mandatory.

Description

If an error occurs executing BACKUP DATA CLOSE SNAPSHOT, use BACKUP DATA DROP SNAPSHOT to remove the latest data snapshot completely from persistence.

Permissions

Privileges needed:
BACKUP OPERATOR or BACKUP ADMIN

Example

Delete the data snapshot in the system database and all the tenant databases.

```
BACKUP DATA FOR FULL SYSTEM DROP SNAPSHOT;
```
Related Information

BACKUP DATA CLOSE SNAPSHOT Statement (Backup and Recovery) [page 704]
BACKUP DATA CREATE SNAPSHOT Statement (Backup and Recovery) [page 706]

4.10.1.40 BACKUP ENCRYPTION ROOT KEYS Statement
(Backup and Recovery)

This statement allows you to back up the encryption Root Keys or LSS backup for the system database or a tenant database.

Syntax

```
BACKUP ENCRYPTION ROOT KEYS [ <root_keytype_list> ] [ FOR <database_name> ] USING <root_key_backup_definition_file>
```

Syntax Elements

```
root_keytype_list ::= root_keytype [ , root_keytype_list ]
```

Specifies the root key type.

If `<root_keytype_list>` is specified (all or any combination of key purposes).

- If the system is connected to KMS:
  - an exception will thrown
- If the system is not connected to KMS:
  - SSFS Format root keys backup will be provided

The following SQL statements are examples of the output file format for the above-mentioned cases; here the LSS-KMS column indicates that the LSS is connected to a KMS (external key management system):

<table>
<thead>
<tr>
<th>SQL</th>
<th>LSS • KMS</th>
<th>LSS</th>
<th>SSFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP ENCRYPTION ROOT KEYS PERSISTENCE...</td>
<td>Throw</td>
<td>SSFS Format root keys backup</td>
<td>SSFS Format root keys backup</td>
</tr>
<tr>
<td>BACKUP ENCRYPTION ROOT KEYS PERSISTENCE, APPLICATION...</td>
<td>Throw</td>
<td>SSFS Format root keys backup</td>
<td>SSFS Format root keys backup</td>
</tr>
</tbody>
</table>
If a `<root_keytype_list>` is not specified:

- If SSFS is used, all root keys will be backed up in an SSFS Format root keys backup.
- If LSS is used, a full Secure Store backup is created (all root keys, KMCs, metadata, etc), which is referred as a LSS Backup.

The following SQL statement illustrates what happens if SSFS or LSS is used:

```sql
SQL | LSS - KMS | LSS | SSFS
---|---|---|---
BACKUP ENCRYPTION ROOT KEYS... | Throw | SSFS Format root keys backup | SSFS Format root keys backup
```

**database_name**

Specifies the name of the tenant database.

To back up the system database, FOR `<database_name>` does not need to be specified.

```sql
root_keytype ::= PERSISTENCE | APPLICATION | BACKUP | LOG
```

`<root_keytype>` can be PERSISTENCE, APPLICATION, BACKUP, or LOG, or a combination of them.

```sql
root_key_backup_definition_file ::= FILE (prefix | path, prefix)
```

Specifies the root key backup definition file. The file is written to the file system on the database back-end.

The name of the backup file is constructed from the following elements:

```sql
<prefix>_<database_name>_rootkeys
```

If `<path>` is not specified, SAP HANA attempts to retrieve the backup location from the configuration parameter `<basepath_rootkeybackup>` in the `<persistence>` section of the global.ini file.

If the basepath_rootkeybackup parameter is not configured, the default backup path `$ {DIR_INSTANCE}/backup/sec` is used.

### Description

**Restriction**

This statement can only be used to create encryption root key or LSS backups in the file system. Use with third-party backup tools is not supported.
The statement must be executed in the system database. The tenant database can be online or offline.

Before the BACKUP ENCRYPTION ROOT KEYS statement is executed, encryption configuration control must be given to the system database for the tenant database. Use the following SQL statement:

```
ALTER SYSTEM ENCRYPTION CONFIGURATION CONTROLLED BY SYSTEM DATABASE
```

**i Note**

If the system database does not own encryption key management control for the tenant database, the statement will fail.

**Backup Password**

A password for the backup of encryption root keys must be set before the backup is created.

To set a password for the system database, use the following SQL statement on the system database:

```
ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD <PASSWORD>
```

To set a password for a tenant database, use the following SQL statement from the system database:

```
ALTER DATABASE <database_name> SET ENCRYPTION ROOT KEYS BACKUP PASSWORD <PASSWORD>
```

**Permissions**

The use of this statement requires the system privilege ENCRYPTION ROOT KEY ADMIN.

For the **system database**, this statement requires one of the following system privileges:

- BACKUP ADMIN
- BACKUP OPERATOR

For a **tenant database**, this statement requires one of the following system privileges:

- DATABASE BACKUP ADMIN
- DATABASE BACKUP OPERATOR

Actions on the backup file on operating system level, for example to copy the file to additional locations, require an operating system user.

**Example**

```
BACKUP ENCRYPTION ROOT KEYS USING FILE ('/backup/HDB/rootkeys/backup001')
```
If SSFS is used, all root keys will be backed up in an SSFS Format root keys backup will be provided. If LSS is used, a full Secure Store backup is created (all root keys, KMCs, metadata, etc). Please note that the statement will provide the backup for the system database as FOR <DATABASE_NAME> is omitted.

```
BACKUP ENCRYPTION ROOT KEYS FOR testDB USING FILE ('backup001')
```

If SSFS is used, all root keys will be backed up in an SSFS Format root keys backup will be provided. If LSS is used, a full Secure Store backup is created (all root keys, KMCs, metadata, etc). For the tenant database testDB, the output file prefix is backup001.

A backup file is generated with the name: backup001_testDB_rootkeys.

### Related Information

- RECOVER ENCRYPTION ROOT KEYS Statement (Backup and Recovery) [page 1079]
- ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 568]
- ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]

**Back Up Root Keys**
**Root Key Backup**

### 4.10.1.41 BACKUP LIST DATA Statement (Backup and Recovery)

Uses the backup catalog to generate a list of data backups that can be used to recover SAP HANA to a specified point in time.

**Syntax**

```
BACKUP LIST DATA [ <for_database> ] <until_definition> [ <system_source_definition> ] [ <path_definition> ] [ <limit_offset> ] |
BACKUP COMPLETE LIST DATA [ <for_database> ] [ <system_source_definition> ] [ <path_definition> ] [ <limit_offset> ] |
```

**Syntax Elements**

- **for_database**
  Specifies the backup catalog for a tenant database.

```
<for_database> ::= FOR <databasename>
```
For example: `BACKUP LIST DATA FOR DB0 UNTIL TIMESTAMP '2020-10-21'`.

Requirements:

- Executed on the system database only.
- The system database must be online, but the tenant database must be offline.

### until_definition

Specifies a point, up to which to return a list of data backups from the backup catalog. For example, backups that could be used for a recovery up to a specified point in time.

```
<until_definition> ::=   UNTIL LOG POSITION <log_pos> AT VOLUME <volume_id>   | UNTIL TIMESTAMP '<timestamp>'
```

### system_source_definition

Identifies the source system.

This option can only be used for a backup catalog from a third-party backup tool.

```
<system_source_definition> ::= USING SOURCE <system_id>
```

### path_definition

Specifies the path to the backup catalog.

```
<path_definition> ::= USING CATALOG BACKINT   | USING CATALOG PATH ( <path> )
```

In the file system, SAP HANA searches the specified locations for the backup catalog in the following order:

<table>
<thead>
<tr>
<th>Path Specified...</th>
<th>Where Searched...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In the SQL statement (optional):</td>
<td>The path specified is searched for a backup catalog.</td>
</tr>
<tr>
<td>USING CATALOG PATH</td>
<td>No subdirectories are searched.</td>
</tr>
<tr>
<td>2. Configured path.</td>
<td>The path to be searched is specified by the following parameter:</td>
</tr>
<tr>
<td></td>
<td>basepath_catalogbackup</td>
</tr>
<tr>
<td></td>
<td>This parameter is in the persistence section of the global.ini parameter file.</td>
</tr>
</tbody>
</table>

### limit_offset

The following options can be used with SAP HANA tenant databases:

- **LIMIT <limit>** restricts the number of elements returned. If **LIMIT <limit>** is not specified, then a default value of 10 is used.
- **OFFSET <offset>** defines the number of data backups to skip at the beginning of the list. If **OFFSET <offset>** is not specified, then no data backups are skipped.

```
<limit_offset> ::= LIMIT <int_const> [OFFSET <int_const>]
```
Description

The most recent backup catalog available is searched at either the default log backup location or at the location specified by `<path_definition>`. You can restrict the search to only a third-party backup tool by using `USING CATALOG BACKINT`.

BACKUP COMPLETE LIST DATA

The `COMPLETE` option provides a list of complete data backups that can be used for a recovery. For this reason, no: `DATABASE RECOVERY OPERATOR` or `DATABASE ADMIN`<until_definition> is used with `BACKUP COMPLETE LIST DATA`. The `COMPLETE` option lists all the data backups from the backup catalog, starting with the most recent. This includes data backups that are not visible as a result of a previous recovery.

Error Handling

The result set of `BACKUP LIST DATA` is written to an XML file named `strategyOutput.xml` in the SAP HANA trace directory.

If an error occurs, `strategyOutput.xml` is deleted.

An SQL error is not always returned. If `strategyOutput.xml` was not created or was rewritten, check the `backup.log` for error messages.

For more information, see `backup.log` in the *SAP HANA Administration Guide (SAP HANA Database Backup and Recovery)*.

Permissions

Privilege needed:

FOR `<DATABASE>`

Related Information

: `DATABASE RECOVERY OPERATOR` or `DATABASE` `backup.log` (SAP HANA Administration Guide (SAP HANA Database Backup and Recovery))
4.10.1.42 CALL Statement (Procedural)

Calls a procedure.

Syntax

CALL <proc_name> ( <param_list> ) [ <hint_clause> ] [ WITH OVERVIEW]
| CALL <sqlscript_lib_member> ( <param_list> )

Syntax Elements

proc_name
Specifies the procedure to be called.

<proc_name> ::= [ <schema_name>.]<identifier>

Optionally, specify a schema name for the identifier.

param_list
Specifies the procedure parameters.

<param_list> ::= <proc_param>[,...]

proc_param
Specifies procedure parameters.

<proc_param> ::=<identifier>
| <string_literal>
| <unsigned_integer>
| <signed_integer>
| <signed_numeric_literal>
| <unsigned_numeric_literal>
| <expressions>

Parameters passed to a procedure are scalar constants and can be passed either as IN, OUT, or INOUT parameters. Scalar parameters are assumed to be NOT NULL. Arguments for IN parameters of table type can either be physical tables or views. The actual value passed for tabular OUT parameters must be ?.

hint_clause
Specifies a hint to use for the CALL.

<hint_clause> ::= WITH HINT ( <hint> [, <hint> ...] )
<hint> ::= { <hint_element> | <routing_hint> | <value_input> } [ CASCADE ]
<hint_element> ::= <hint_name> from public hint list [(<hint_argument_list>)]
<hint_argument_list> ::= <hint_argument> [, <hint_argument> ...]
For a list of available hints by category, see HINT Details [page 1134].

**CASCADE** Allows propagation of a hint into internal SQL statements within a procedure. The HINT is applied to both the CALL statement itself and any internal SQLs. If CASCADE is used for non-CALL statements, a FEATURE NOT SUPPORTED message appears. CASCADE is only supported in CALL statements and cannot propagate hints into user-defined functions. When there are nested CALL statements, CASCADE hints are also propagated into the nested procedures.

**WITH OVERVIEW**

Defines that the result of a procedure call is stored directly into a physical table.

Calling a procedure WITH OVERVIEW returns one result set that contains the information about which table contains the result of which table output variable. Scalar outputs are returned as regular scalar output parameters. When you pass existing tables to the output parameters, WITH OVERVIEW inserts the result set tuples of the procedure into the provided tables. When you pass NULL to the output parameters, temporary tables holding the result sets are generated. These tables are dropped automatically once the database session is closed.

**sqlscript_lib_member**

Calls a SQLScript library member procedure.

When calling a SQLScript library member procedure, the WITH option is not supported. Also, there is no EXPLAIN PLAN or QUERY EXPORT support when calling a SQLScript library member procedure.

For more information about SQLScript user-defined libraries, see the SAP HANA SQLScript Reference Guide.

**Description**

Calls a procedure defined using the CREATE PROCEDURE statement.

CALL conceptually returns list of result sets with one entry for every tabular result. An iterator can be used to iterate over these results sets. For each result set you can iterate over the result table in the same way as for query results. SQL statements that are not assigned to any table variable in the procedure body are
added as result sets at the end of the list of result sets. The type of the result structures is determined during compilation time, but it is not visible in the signature of the procedure.

When executed by the client, the CALL syntax behaves in a way consistent with the SQL standard semantics; for example, Java clients can call a procedure using a JDBC CallableStatement. Scalar output variables are a scalar value that can be retrieved from the callable statement directly.

Unquoted identifiers are implicitly treated as upper case. Quoting identifiers respect capitalization and allow for using white spaces which are normally not allowed in SQL identifiers.

**Examples**

Call the getOutput procedure in debug mode.

```sql
CALL getOutput (1000, 'EUR', NULL, NULL) IN DEBUG MODE;
```

Call the procedure getOutput using the WITH OVERVIEW option.

```sql
CALL getOutput(1000, 'EUR', ot_publishers, NULL) WITH OVERVIEW;
```

The following example creates a library with one member procedure and then calls the member procedure directly using the CALL statement:

```sql
CREATE LIBRARY MyLib AS BEGIN
   PUBLIC PROCEDURE MemberProc(IN i INT, OUT tv TABLE(Col1 NVARCHAR(10))) AS BEGIN
      tv = SELECT :i * 100 AS Col1 FROM DUMMY;
   END;
END;
CALL MyLib:MemberProc(1, ?);
```

**Related Information**

- Table Variable Type Definition
- User-Defined Libraries
- SAP HANA SQLScript Reference
- SQL Notation Conventions [page 30]
- Data Types [page 33]
- CREATE PROCEDURE Statement (Procedural) [page 776]
4.10.1.43 COMMENT ON Statement (Data Definition)

Adds a comment to an object in the database, or removes an existing comment.

Syntax

```
COMMENT ON <object_type> <object_name> IS <comment>
```

Syntax Elements

- `object_type`
  Specifies the type of object to add the comment on.

  `<object_type> ::= TABLE | VIEW | COLUMN | USER | ROLE | USERGROUP | SCHEDULER JOB | CERTIFICATE | PUBLIC KEY`

- `object_name`
  Specifies the name of the object to add a commented on, with optional schema name.

  `<object_name> ::= [<schema_name>.]<identifier>`

- `comment`
  Specifies the comment for the object.

  `<comment> ::= <string_literal> | NULL`

  If NULL is specified, then any existing comment is dropped.

Description

Comments are a useful way to record an intuitive description of objects in the database such as tables, columns, and users. You can also remove comments.
Permissions

To add or remove comments on an object, you must have the following object-specific privileges:

- For tables, views, and view columns: ALTER object privilege on the table or view
- For users: OPERATOR object privilege on the user group of the user
- For roles: ROLE ADMIN if the role is not in a role group, otherwise either the GRANTOR or OPERATOR object privilege on the role group that role is assigned
- For user groups: OPERATOR object privilege on the user group
- For scheduler jobs: ALTER object privilege on the scheduler job
- For role groups: GRANTOR or OPERATOR object privilege on the role group
- For certificates and public keys: CERTIFICATE ADMIN system privilege

Examples

Comment on a table

Create a table and add a comment on it.

```sql
CREATE ROW TABLE COMMENT_ON_EX(A INT);
COMMENT ON TABLE COMMENT_ON_EX IS 'Used for comment on examples';
```

Select the table from the TABLES system table to view the comment.

```sql
SELECT * FROM TABLES WHERE table_name='COMMENT_ON_EX';
```

Comment on a table column

Comment on column A of the COMMENT_ON_EX table from example 1.

```sql
COMMENT ON COLUMN COMMENT_ON_EX.A IS 'This is an example column comment';
```

Select the column description from the TABLE_COLUMNS system table.

```sql
SELECT * FROM TABLE_COLUMNS WHERE table_name='COMMENT_ON_EX';
```

Comment on a view and view column

Create a view, a view comment, and view column comment.

```sql
CREATE VIEW COMMENT_ON_EX_VIEW AS SELECT * FROM COMMENT_ON_EX;
COMMENT ON VIEW COMMENT_ON_EX_VIEW IS 'This is a view comment';
COMMENT ON COLUMN COMMENT_ON_EX_VIEW.A IS 'This is a view column comment';
```

Select the view comment and view column comments from the relevant system tables.

```sql
SELECT * FROM VIEWS WHERE view_name='COMMENT_ON_EX_VIEW';
SELECT * FROM VIEW_COLUMNS WHERE view_name='COMMENT_ON_EX_VIEW';
```

Comment on a scheduler job
Create a scheduled job and a scheduler job comment.

```sql
CREATE SCHEDULER JOB retention_job CRON '** * * * 1 30 01' ENABLE PROCEDURE
RETENTION_JOB PARAMETERS DAYS=14
COMMENT ON SCHEDULER JOB retention_job IS 'This job implements a retention
policy';
```

Remove comments

Drop a view comment and a view column comment.

```sql
COMMENT ON VIEW COMMENT_ON_EX_View IS NULL;
COMMENT ON COLUMN COMMENT_ON_EX_View.A IS NULL;
```

Related Information

TABLES System View [page 1669]
TABLE_COLUMNs System View [page 1675]
VIEWS System View [page 1702]
VIEW_COLUMNs System View [page 1705]

4.10.1.44 COMMIT Statement (Transaction Management)

Makes changes to the database permanent, or terminates a user-defined transaction.

**Syntax**

```sql
COMMIT
```

**Description**

The system supports transactional consistency that guarantees the current job to be either completely applied
to the system or disposed of.

If a user wants to apply the current job to the system persistently, then the user should issue a COMMIT
command. If a COMMIT command is issued and successfully processed, then any change on the system that
the current transaction has done is applied to the system and the change is visible to other jobs that start later.
A job that has already been committed a via COMMIT command cannot be reverted.

In a distributed system, a standard two-phase-commit protocol is complied. In the first phase, transaction
coordinator consults every participant whether if it is ready to commit, and sends the result to the participants
in the second phase. The COMMIT command only works with an autocommit-disabled session.
Example

```sql
COMMIT;
```

### 4.10.1.45 CONNECT Statement (Session Management)

Connects to a database instance.

**Syntax**

```sql
CONNECT      { <user_name> PASSWORD <password>     
  WITH SAML ASSERTION <xml>     
  WITH JWT <token> }
```

**Syntax Elements**

- **user_name**
  Specifies the user name for the connection.

  ```sql
  <user_name> ::= <unicode_name>
  ```

- **WITH SAML ASSERTION xml**
  Specifies a SAML assertion to use when connecting to the database.

  ```sql
  WITH SAML ASSERTION <xml>
  <xml> ::= <string_literal>
  ```

- **WITH JWT token**
  Specifies a JWT token to use when connecting to the database.

  ```sql
  WITH JWT <token>
  <token> ::= <string_literal>
  ```

**Description**

Connects to a database instance using either a user name and password pair, an SAML or JWT assertion, or a JWT token for identification.

Automatic LDAP user creation is not supported with the CONNECT statement.
Example

Create a SAML provider named ac_saml_provider specifying a subject and issuer for ACompany.

```sql
CREATE SAML PROVIDER ac_saml_provider
   WITH SUBJECT 'CN = wiki.detroit.ACompany.corp,OU = ACNet,O = ACompany,C = EN'
   ISSUER 'E = John.Do@acompany.com,CN = ACNetCA,OU = ACNet,O = ACompany,C = EN';
```

Create a user called new_user with the password Password1. The user can connect to the system using the given password and with assertion of the SAML provider ac_saml_provider.

```sql
CREATE USER new_user PASSWORD Password1 WITH IDENTITY ANY FOR SAML PROVIDER ac_saml_provider;
```

Connect to the database with the user name new_user and password Password1.

```sql
CONNECT new_user PASSWORD Password1;
```

4.10.1.46  CREATE AUDIT POLICY Statement (Access Control)

Creates an audit policy.

Syntax

```sql
CREATE AUDIT POLICY <policy_name> [ FOR <database_name> ] AUDITING
   <audit_status> <audit_actions>
   LEVEL <audit_level> [ <opt_audit_trail_type> ]
```

Syntax Elements

- **policy_name**
  Specifies the name of the audit policy to be created.

- **database_name**
  Defines the name of the tenant the audit policy applies to. Only one tenant can be specified for each audit policy and this option can only be used when creating audit policies on SYSTEMDB.

- **audit_status**
  Defines whether successful, unsuccessful, or all executions of the specified audit actions are audited.
audit_actions
Specifies the audit actions for the audit policy.

<audit_actions> ::= { <actions_for_principals> | <audit_action_list> | <target_audit_action_list> | <schema_audit_action_list> }

actions_for_principals
Specifies that commands executed by a user or a set of users or all users except the named set of users are audited.

<actions_for_user> ::= ACTIONS [ EXCEPT ] FOR <user_name> [, <user_name> [...] ]

user_name
Specifies the username of the user to be audited by the audit policy.

<user_name> ::= <identifier>

audit_action_list
Audits specific system actions, optionally limited by <principal_list>.

<audit_action_list> ::= <audit_action_name> [, <audit_action_name> [...] ] [ [ EXCEPT ] FOR <principal_list>]

For the complete list of possible values for <audit_action_name>, see the table of audit actions provided in the Description section of this topic.

principal_list
Specifies the principals whose actions or whose members’ actions are to be audited or excluded from auditing. Principals can be users, usergroups, or a set of users and usergroups.

<principal_list> ::= { <user_name> [, <user_name> [...] ] |
PRINCIPALS [ { USER <user_name> | USERGROUP <usergroup_name> } [...] ] }

target_audit_action_list
Audits actions on a database object or a set of objects. Optionally, this auditing can be limited to a user or a set of users or audited for all users except for the given set of users.

<target_audit_action_list> ::= <target_audit_action_entry> [ [ EXCEPT ] <principal_list> ]

target_audit_action_entry
Only tables, views, procedures, and schemas can be specified.

<target_audit_action_entry> ::= <target_audit_action_name> [, <target_audit_action_name> [...] ]
In the case of `<schema_name>.*`, all objects that are stored in this schema and can be combined with the specified `<target_audit_action_name>` values are audited. Synonyms and sequences cannot be selected as objects for audit policies. Only specified `<target_audit_action_name>` values can be combined with an object.

### target_audit_action_name

```sql
<target_audit_action_name> ::=   { INSERT | UPDATE | DELETE | SELECT | EXECUTE }
```

### audit_object_name

```sql
<audit_object_name> ::=    { <table_name> | <view_name> | <procedure_name> | <schema_name>. }
```

- `<table_name>`
  ```sql
  ::= [ <schema_name>.]<identifier>
  ```

- `<view_name>`
  ```sql
  ::= [ <schema_name>.]<identifier>
  ```

- `<procedure_name>`
  ```sql
  ::= [ <schema_name>.]<identifier>
  ```

- `<schema_name>`
  ```sql
  ::= <unicode_name>
  ```

Specifies a database object for the target audit action.

### schema_audit_action_list

```sql
<schema_audit_action_list> ::= <schema_audit_action_entry> [ [ EXCEPT ] FOR <user_name>[, <user_name> {,...} ] ]
```

### schema_audit_action_entry

```sql
<schema_audit_action_entry> ::=<schema_audit_action_name>[, ...]
 [ ON SCHEMA <audit_schema_name>[, <audit_schema_name> [,,...] ] ]
```

### schema_audit_action_name

```sql
<schema_audit_action_name> ::=   { ALTER TABLE
                      CREATE TABLE
                      DROP TABLE
                      RENAME TABLE
                      CREATE PROCEDURE
                      DROP PROCEDURE
                      ALTER PROCEDURE
                      CREATE FUNCTION
                      DROP FUNCTION
                      ALTER FUNCTION
                      CREATE TRIGGER
                      DROP TRIGGER
                      CREATE VIEW
```
audit_level

Assigns an audit policy to an audit level.

```
<audit_level> ::= EMERGENCY | ALERT | CRITICAL | WARNING | INFO
```

The levels are listed above in decreasing order of importance.

opt_audit_trail_type

Specifies the audit trail target type(s) for the audit policy.

```
<opt_audit_trail_type> ::= TRAIL TYPE <audit_trail_type_list>
<audit_trail_type_list> ::= <audit_trail_type_name>, ...
<audit_trail_type_name> ::= TABLE RETENTION <duration>
<duration> ::= <integer>
```

RETENTION duration Specifies the length of time the policy is retained, expressed in days. A minimum allowed value can be configured with the configuration parameter [auditing configuration] minimum_retention_period. The default is 7 days.
The CREATE AUDIT POLICY statement creates a new audit policy, which monitors when specified audit actions occur. Only database users with the AUDIT ADMIN privilege can create an audit policy. An audit policy name must be unique.

New audit policies are disabled by default and must be enabled before the audit actions begin. The configuration parameter global_auditing_state must also be set to true.

An audit policy can contain only one of the following:

- non-restricted auditing for n (>=1) users
- auditing for actions not restricted to objects
- auditing for actions that are restricted to objects

For the last two alternatives listed, an optional restriction for user(s) is available.

Audit actions related to tenant databases (like CREATE/ALTER/DROP/START/STOP DATABASE) can only be specified in the SYSTEMDB of a tenant (database) system.

### Audit Actions

One or more audit trail targets can be specified for an audit policy at the time of creation or after creation. The allowed audit trail targets are:

- **SYSLOG**: uses the system syslog.
- **TABLE**: stores audit information in database table. The audit log is accessible using AUDIT_LOG system view.
- **CSV**: stores audit information as comma-separated values in a text file. Use only for testing purposes.

A subset of audit actions are allowed in policies created on SYSTEMDB for a tenant using the `FOR <database_name>` option. Audit policies created for a tenant on SYSTEMDB cannot be altered or dropped from within that tenant. It must be done on SYSTEMDB.

<table>
<thead>
<tr>
<th>Audit Action Name</th>
<th>Group Number</th>
<th>Audit Operation</th>
<th>SYSTEMDB Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANCEL SESSION</td>
<td>1 - Session Management and System Configuration</td>
<td>Audits the use of ALTER SYSTEM CANCEL SESSION to cancel sessions.</td>
<td>No</td>
</tr>
<tr>
<td>CONNECT</td>
<td>1 - Session Management and System Configuration</td>
<td>Audits the creation of a user connection to the database.</td>
<td>Yes</td>
</tr>
<tr>
<td>DISCONNECT SESSION</td>
<td>1 - Session Management and System Configuration</td>
<td>Audits the use of ALTER SYSTEM DISCONNECT SESSION to disconnect sessions.</td>
<td>No</td>
</tr>
<tr>
<td>STOP SERVICE</td>
<td>1 - Session Management and System Configuration</td>
<td>Audits the stopping of services.</td>
<td>No</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>SYSTEM CONFIGURATION CHANGE</td>
<td>1 - Session Management and System Configuration</td>
<td>Audits changes to the system configuration, such as the INIFILE.</td>
<td>Yes</td>
</tr>
<tr>
<td>VALIDATE USER</td>
<td>1 - Session Management and System Configuration</td>
<td>Audits the validation of a user’s credentials.</td>
<td>Yes</td>
</tr>
<tr>
<td>GRANT ANY</td>
<td>2 - Granting and Revoking of Authorization</td>
<td>Audits the granting of privileges, structured privileges, or roles to users or roles.</td>
<td>No</td>
</tr>
<tr>
<td>GRANT APPLICATION PRIVILEGE</td>
<td>2 - Granting and Revoking of Authorization</td>
<td>Audits the granting of application privileges to users or roles.</td>
<td>No</td>
</tr>
<tr>
<td>GRANT PRIVILEGE</td>
<td>2 - Granting and Revoking of Authorization</td>
<td>Audits the granting of privileges to users or roles.</td>
<td>No</td>
</tr>
<tr>
<td>GRANT ROLE</td>
<td>2 - Granting and Revoking of Authorization</td>
<td>Audits the granting of roles to users or roles.</td>
<td>No</td>
</tr>
<tr>
<td>GRANT STRUCTURED PRIVILEGE</td>
<td>2 - Granting and Revoking of Authorization</td>
<td>Audits the granting of structured/analytical privileges to users or roles.</td>
<td>No</td>
</tr>
<tr>
<td>REVOKE ANY</td>
<td>2 - Granting and Revoking of Authorization</td>
<td>Audits the revoking of privileges, structured privileges, or roles from users or roles.</td>
<td>No</td>
</tr>
<tr>
<td>REVOKE APPLICATION PRIVILEGE</td>
<td>2 - Granting and Revoking of Authorization</td>
<td>Audits the revoking of application privileges from users or roles.</td>
<td>No</td>
</tr>
<tr>
<td>REVOKE PRIVILEGE</td>
<td>2 - Granting and Revoking of Authorization</td>
<td>Audits the revoking of privileges from users or roles.</td>
<td>No</td>
</tr>
<tr>
<td>REVOKE ROLE</td>
<td>2 - Granting and Revoking of Authorization</td>
<td>Audits the revoking of roles from users or roles.</td>
<td>No</td>
</tr>
<tr>
<td>REVOKE STRUCTURED PRIVILEGE</td>
<td>2 - Granting and Revoking of Authorization</td>
<td>Audits the revoking of structured/analytical privileges from users or roles.</td>
<td>No</td>
</tr>
<tr>
<td>DELETE</td>
<td>3 - Data Query and Manipulation</td>
<td>Audits the deletion of rows from tables/views and the truncation of tables. Allows the specification of target objects.</td>
<td>No</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>3 - Data Query and Manipulation</td>
<td>Audits the execution of procedure calls. Allows the specification of target objects.</td>
<td>No</td>
</tr>
<tr>
<td>INSERT</td>
<td>3 - Data Query and Manipulation</td>
<td>Audits the use of INSERT/REPLACE/UPSERT statements on tables and views. Allows the specification of target objects.</td>
<td>No</td>
</tr>
<tr>
<td>SELECT</td>
<td>3 - Data Query and Manipulation</td>
<td>Audits the use of SELECT statements on tables and views. Allows specification of target objects.</td>
<td>No</td>
</tr>
<tr>
<td>UPDATE</td>
<td>3 - Data Query and Manipulation</td>
<td>Audits the use of UPDATE/REPLACE/UPSERT statements on tables and views. Allows specification of target objects.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER ROLE</td>
<td>4 - User and Role Management</td>
<td>Audits the alteration of roles.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER USER</td>
<td>4 - User and Role Management</td>
<td>Audits the alteration of users.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER USERGROUP</td>
<td>4 - User and Role Management</td>
<td>Audits the alteration of user groups.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE ROLE</td>
<td>4 - User and Role Management</td>
<td>Audits the creation of roles.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE USER</td>
<td>4 - User and Role Management</td>
<td>Audits the creation of users.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE USERGROUP</td>
<td>4 - User and Role Management</td>
<td>Audits the creation of user groups.</td>
<td>No</td>
</tr>
<tr>
<td>DROP ROLE</td>
<td>4 - User and Role Management</td>
<td>Audits the dropping of roles.</td>
<td>No</td>
</tr>
<tr>
<td>DROP USER</td>
<td>4 - User and Role Management</td>
<td>Audits the dropping of users.</td>
<td>No</td>
</tr>
<tr>
<td>DROP USERGROUP</td>
<td>4 - User and Role Management</td>
<td>Audits the dropping of user groups.</td>
<td>No</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>ALTER STRUCTURED PRIVILEGE</td>
<td>7 - Structured Privilege Management</td>
<td>Audits the altering of the structured/analytical privilege.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE STRUCTURED PRIVILEGE</td>
<td>7 - Structured Privilege Management</td>
<td>Audits the creation of the structured/analytical privilege.</td>
<td>No</td>
</tr>
<tr>
<td>DROP STRUCTURED PRIVILEGE</td>
<td>7 - Structured Privilege Management</td>
<td>Audits the dropping of the structured/analytical privilege.</td>
<td>No</td>
</tr>
<tr>
<td>SET SYSTEM LICENSE</td>
<td>8 - License Installation</td>
<td>Audits the installation of system licenses.</td>
<td>Yes</td>
</tr>
<tr>
<td>UNSET SYSTEM LICENSE</td>
<td>9 - License Deletion</td>
<td>Audits the deletion of licenses.</td>
<td>Yes</td>
</tr>
<tr>
<td>ACTIVATE REPOSITORY CONTENT</td>
<td>10 - Repository Privilege Management</td>
<td>Audits the activation of repository design time objects.</td>
<td>No</td>
</tr>
<tr>
<td>EXPORT REPOSITORY CONTENT</td>
<td>10 - Repository Privilege Management</td>
<td>Audits the export of repository design time objects.</td>
<td>No</td>
</tr>
<tr>
<td>IMPORT REPOSITORY CONTENT</td>
<td>10 - Repository Privilege Management</td>
<td>Audits the import of repository design time objects.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER FULLTEXT INDEX</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of full-text indexes.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER FUNCTION</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of functions.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER GEOCODE INDEX</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of geo-code indexes.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER INDEX</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of indexes.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER PROCEDURE</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of procedures.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER SCHEDULER JOB</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of scheduler jobs.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER SEQUENCE</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of sequences.</td>
<td>No</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>ALTER STATISTICS</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of statistics.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER TABLE</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of tables.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER VIEW</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of views.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER WORKLOAD CLASS</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of workload classes.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER WORKLOAD MAPPING</td>
<td>11 - Data Definition</td>
<td>Audits the alteration of workload mappings.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE FULLTEXT INDEX</td>
<td>11 - Data Definition</td>
<td>Audits the creation of full text indexes.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE FUNCTION</td>
<td>11 - Data Definition</td>
<td>Audits the creation of functions.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE GEOCODE INDEX</td>
<td>11 - Data Definition</td>
<td>Audits the creation of geocode indexes.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE GRAPH WORKSPACE</td>
<td>11 - Data Definition</td>
<td>Audits the creation of graph workspaces.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE INDEX</td>
<td>11 - Data Definition</td>
<td>Audits the creation of indexes.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE PROCEDURE</td>
<td>11 - Data Definition</td>
<td>Audits the creation of procedures.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE SCHEDULER JOB</td>
<td>11 - Data Definition</td>
<td>Audits the creation of scheduler jobs.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE SEQUENCE</td>
<td>11 - Data Definition</td>
<td>Audits the creation of sequences.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE STATISTICS</td>
<td>11 - Data Definition</td>
<td>Audits the creation of statistics.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE SYNONYM</td>
<td>11 - Data Definition</td>
<td>Audits the creation of synonyms.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE SCHEMA</td>
<td>11 - Data Definition</td>
<td>Audits the creation of schemas.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE TABLE</td>
<td>11 - Data Definition</td>
<td>Audits the creation of tables.</td>
<td>No</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------</td>
<td>------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>CREATE TRIGGER</td>
<td>11 - Data Definition</td>
<td>Audits the creation of triggers.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE VIEW</td>
<td>11 - Data Definition</td>
<td>Audits the creation of views.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE WORKLOAD CLASS</td>
<td>11 - Data Definition</td>
<td>Audits the creation of workload classes.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE WORKLOAD MAPPING</td>
<td>11 - Data Definition</td>
<td>Audits the creation of workload mappings.</td>
<td>No</td>
</tr>
<tr>
<td>DROP FULLTEXT INDEX</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of full text indexes.</td>
<td>No</td>
</tr>
<tr>
<td>DROP FUNCTION</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of functions.</td>
<td>No</td>
</tr>
<tr>
<td>DROP GEOCODE INDEX</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of geo-code indexes.</td>
<td>No</td>
</tr>
<tr>
<td>DROP GRAPH WORKSPACE</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of graph workspaces.</td>
<td>No</td>
</tr>
<tr>
<td>DROP INDEX</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of indexes.</td>
<td>No</td>
</tr>
<tr>
<td>DROP PROCEDURE</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of procedures.</td>
<td>No</td>
</tr>
<tr>
<td>DROP SCHEDULER JOB</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of scheduler jobs.</td>
<td>No</td>
</tr>
<tr>
<td>DROP SCHEMA</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of schemas.</td>
<td>No</td>
</tr>
<tr>
<td>DROP SEQUENCE</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of sequences.</td>
<td>No</td>
</tr>
<tr>
<td>DROP STATISTICS</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of statistics.</td>
<td>No</td>
</tr>
<tr>
<td>DROP SYNONYM</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of synonyms.</td>
<td>No</td>
</tr>
<tr>
<td>DROP TABLE</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of tables.</td>
<td>No</td>
</tr>
<tr>
<td>DROP TRIGGER</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of triggers.</td>
<td>No</td>
</tr>
<tr>
<td>DROP VIEW</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of views.</td>
<td>No</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>DROP WORKLOAD CLASS</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of workload classes.</td>
<td>No</td>
</tr>
<tr>
<td>DROP WORKLOAD MAPPING</td>
<td>11 - Data Definition</td>
<td>Audits the dropping of workload mappings.</td>
<td>No</td>
</tr>
<tr>
<td>REFRESH STATISTICS</td>
<td>11 - Data Definition</td>
<td>Audits the refreshing of statistics.</td>
<td>No</td>
</tr>
<tr>
<td>RENAME COLUMN</td>
<td>11 - Data Definition</td>
<td>Audits the renaming of columns</td>
<td>No</td>
</tr>
<tr>
<td>RENAME INDEX</td>
<td>11 - Data Definition</td>
<td>Audits the renaming of indexes</td>
<td>No</td>
</tr>
<tr>
<td>RENAME TABLE</td>
<td>11 - Data Definition</td>
<td>Audits the renaming of tables</td>
<td>No</td>
</tr>
<tr>
<td>ACTIONS</td>
<td>12 - All Actions</td>
<td>Audits all user-triggered database actions. Used for specific users.</td>
<td>No</td>
</tr>
<tr>
<td>BACKUP CATALOG DELETE</td>
<td>13 - Backup and Recovery</td>
<td>Audits the deletion of entries in the backup catalog.</td>
<td>No</td>
</tr>
<tr>
<td>BACKUP DATA</td>
<td>13 - Backup and Recovery</td>
<td>Audits the backing up of data.</td>
<td>No</td>
</tr>
<tr>
<td>RECOVER DATA</td>
<td>13 - Backup and Recovery</td>
<td>Audits the recovery of data.</td>
<td>No</td>
</tr>
<tr>
<td>ACTIVATE KEY MANAGEMENT CONFIGURATION</td>
<td>14 - Volume Encryption</td>
<td>Audits the activation of key management configuration.</td>
<td>No</td>
</tr>
<tr>
<td>ADD KEY MANAGEMENT CONFIGURATION</td>
<td>14 - Volume Encryption</td>
<td>Audits the addition of key management configuration.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER APPLICATION ENCRYPTION</td>
<td>14 - Volume Encryption</td>
<td>Audits the alteration of application encryption keys.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER APPLICATION ENCRYPTION ROOT KEY</td>
<td>14 - Volume Encryption</td>
<td>Audits the alteration of application encryption root keys.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER BACKUP ENCRYPTION</td>
<td>14 - Volume Encryption</td>
<td>Audits the alteration of backup encryption status.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER BACKUP ENCRYPTION ROOT KEY</td>
<td>14 - Volume Encryption</td>
<td>Audits the alteration of backup encryption root keys.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER KEY MANAGEMENT CONFIGURATION</td>
<td>14 - Volume Encryption</td>
<td>Audits the alteration of key management configuration.</td>
<td>No</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>ALTER LOG ENCRYPTION</td>
<td>14 - Volume Encryption</td>
<td>Audits the alteration of log encryption status.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER LOG ENCRYPTION ROOT KEY</td>
<td>14 - Volume Encryption</td>
<td>Audits the alteration of log encryption root keys.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER PERSISTENCE ENCRYPTION</td>
<td>14 - Volume Encryption</td>
<td>Audits the alteration of database persistence encryption status and page encryption keys.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER PERSISTENCE ENCRYPTION ROOT KEY</td>
<td>14 - Volume Encryption</td>
<td>Altering database persistence encryption root keys.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER ROOT KEYS BACKUP PASSWORD</td>
<td>14 - Volume Encryption</td>
<td>Audits the alteration of the backup password used to protect backup root keys.</td>
<td>Yes</td>
</tr>
<tr>
<td>DROP KEY MANAGEMENT CONFIGURATION</td>
<td>14 - Volume Encryption</td>
<td>Audits the dropping of key management configuration.</td>
<td>No</td>
</tr>
<tr>
<td>ENCRYPTION CONFIG CONTROL</td>
<td>14 - Volume Encryption</td>
<td>Audits the alteration of the database (system or local tenant database) controlling the encryption configuration.</td>
<td>Yes</td>
</tr>
<tr>
<td>TENANT BACKUP ENCRYPTION</td>
<td>14 - Volume Encryption</td>
<td>Audits the switching on or off of the backup encryption for the tenant.</td>
<td>Yes</td>
</tr>
<tr>
<td>TENANT LOG ENCRYPTION</td>
<td>14 - Volume Encryption</td>
<td>Audits the switching on or off of the log encryption for the tenant.</td>
<td>Yes</td>
</tr>
<tr>
<td>TENANT PERSISTENCE ENCRYPTION</td>
<td>14 - Volume Encryption</td>
<td>Audits the switching on or off of the data volume encryption for the tenant.</td>
<td>No</td>
</tr>
<tr>
<td>TENANT ROOT KEYS BACKUP PASSWORD</td>
<td>14 - Volume Encryption</td>
<td>Audits the setting the root key backup password.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE DATABASE¹</td>
<td>16 - Operations on Multitenant Database Containers</td>
<td>Creating a database. Available only for the SYSTEMDB in a multitenant database system.</td>
<td>No</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>ALTER DATABASE¹</td>
<td>16 - Operations on Multitenant Database Containers</td>
<td>Altering a database. Available only for the SYSTEMDB in a multitenant database system.</td>
<td>No</td>
</tr>
<tr>
<td>DROP DATABASE¹</td>
<td>16 - Operations on Multitenant Database Containers</td>
<td>Dropping a database. Available only for the SYSTEMDB in a multitenant database system.</td>
<td>No</td>
</tr>
<tr>
<td>RENAME DATABASE¹</td>
<td>16 - Operations on Multitenant Database Containers</td>
<td>Renaming a database. Available only for the SYSTEMDB in a multitenant database system.</td>
<td>No</td>
</tr>
<tr>
<td>START DATABASE¹</td>
<td>16 - Operations on Multitenant Database Containers</td>
<td>Starting a database. Available only for the SYSTEMDB in a multitenant database system.</td>
<td>No</td>
</tr>
<tr>
<td>STOP DATABASE¹</td>
<td>16 - Operations on Multitenant Database Containers</td>
<td>Stopping a database. Available only for the SYSTEMDB in a multitenant database system.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER ADAPTER</td>
<td>17 - Data Provisioning</td>
<td>Audits the alteration of adapters.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER AGENT</td>
<td>17 - Data Provisioning</td>
<td>Audits the alteration of agents.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER REMOTE SOURCE</td>
<td>17 - Data Provisioning</td>
<td>Audits the alteration of remote sources.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER REMOTE SUBSCRIPTION</td>
<td>17 - Data Provisioning</td>
<td>Audits the alteration of remote subscriptions.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE ADAPTER</td>
<td>17 - Data Provisioning</td>
<td>Audits the creation of adapters.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE AGENT</td>
<td>17 - Data Provisioning</td>
<td>Audits the creation of agents.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE AGENT GROUP</td>
<td>17 - Data Provisioning</td>
<td>Audits the creation of agent groups.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE REMOTE SOURCE</td>
<td>17 - Data Provisioning</td>
<td>Audits the creation of remote sources.</td>
<td>No</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>CREATE REMOTE SUBSCRIPTION</td>
<td>17 - Data Provisioning</td>
<td>Audits the creation of remote subscriptions.</td>
<td>No</td>
</tr>
<tr>
<td>DROP ADAPTER</td>
<td>17 - Data Provisioning</td>
<td>Audits the dropping of adapters.</td>
<td>No</td>
</tr>
<tr>
<td>DROP AGENT</td>
<td>17 - Data Provisioning</td>
<td>Audits the dropping of agents.</td>
<td>No</td>
</tr>
<tr>
<td>DROP AGENT GROUP</td>
<td>17 - Data Provisioning</td>
<td>Audits the dropping of agent groups.</td>
<td>No</td>
</tr>
<tr>
<td>DROP REMOTE SOURCE</td>
<td>17 - Data Provisioning</td>
<td>Audits the dropping of remote sources.</td>
<td>No</td>
</tr>
<tr>
<td>DROP REMOTE SUBSCRIPTION</td>
<td>17 - Data Provisioning</td>
<td>Audits the dropping of remote subscriptions.</td>
<td>No</td>
</tr>
<tr>
<td>PROCESS REMOTE SUBSCRIPTION EXCEPTION</td>
<td>17 - Data Provisioning</td>
<td>Audits the processing of remote subscription exceptions.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER EXTENDED STORAGE</td>
<td>18 - SAP HANA Dynamic Tiering</td>
<td>Audits the altering of extended store.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE EXTENDED STORAGE</td>
<td>18 - SAP HANA Dynamic Tiering</td>
<td>Audits the creation of extended store.</td>
<td>No</td>
</tr>
<tr>
<td>DROP EXTENDED STORAGE</td>
<td>18 - SAP HANA Dynamic Tiering</td>
<td>Audits the dropping of extended store.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER PSE</td>
<td>19 - Certificates and PSE Store</td>
<td>Audits the alteration of PSEs.</td>
<td>Yes</td>
</tr>
<tr>
<td>CREATE CERTIFICATE</td>
<td>19 - Certificates and PSE Store</td>
<td>Audits the creation of certificates.</td>
<td>Yes</td>
</tr>
<tr>
<td>CREATE PSE</td>
<td>19 - Certificates and PSE Store</td>
<td>Audits the creation of PSEs.</td>
<td>Yes</td>
</tr>
<tr>
<td>CREATE PUBLIC KEY</td>
<td>19 - Certificates and PSE Store</td>
<td>Audits the creation of public keys.</td>
<td>Yes</td>
</tr>
<tr>
<td>DROP CERTIFICATE</td>
<td>19 - Certificates and PSE Store</td>
<td>Audits the dropping of certificates.</td>
<td>Yes</td>
</tr>
<tr>
<td>DROP PSE</td>
<td>19 - Certificates and PSE Store</td>
<td>Audits the dropping of PSEs.</td>
<td>Yes</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>DROP PUBLIC KEY</td>
<td>19 - Certificates and PSE Store</td>
<td>Audits the dropping of public keys.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER JWT PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the alteration of JWT providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER LDAP PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the alteration of LDAP providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER SAML PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the alteration of SAML providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER X509 PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the alteration of X509 providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>CREATE JWT PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the creation of JWT providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>CREATE LDAP PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the creation of LDAP providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>CREATE SAML PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the creation of SAML providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>CREATE X509 PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the creation of X509 providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>DROP JWT PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the dropping of JWT providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>DROP LDAP PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the dropping of LDAP providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>DROP SAML PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the dropping of SAML providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>DROP X509 PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the dropping of X509 providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>VALIDATE LDAP PROVIDER</td>
<td>20 - Authentication Provider Management</td>
<td>Audits the refreshing of LDAP providers.</td>
<td>Yes</td>
</tr>
<tr>
<td>ALTER CLIENTSIDE ENCRYPTION COLUMN KEY</td>
<td>21 - Client-side Encryption</td>
<td>Audits the alteration of a column encryption key (CEK).</td>
<td>No</td>
</tr>
<tr>
<td>ALTER CLIENTSIDE ENCRYPTION KEYPAIR</td>
<td>21 - Client-side Encryption</td>
<td>Audits the alteration of a client-side encryption key pair.</td>
<td>No</td>
</tr>
<tr>
<td>Audit Action Name</td>
<td>Group Number</td>
<td>Audit Operation</td>
<td>SYSTEMDB Supported</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>CREATE CLIENTSIDE ENCRYPTION COLUMN KEY</td>
<td>21 - Client-side Encryption</td>
<td>Audits the creation of a CEK.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE CLIENTSIDE ENCRYPTION KEYPAIR</td>
<td>21 - Client-side Encryption</td>
<td>Audits the creation of a client-side encryption key pair.</td>
<td>No</td>
</tr>
<tr>
<td>DROP CLIENTSIDE ENCRYPTION COLUMN KEY</td>
<td>21 - Client-side Encryption</td>
<td>Audits the dropping of a CEK.</td>
<td>No</td>
</tr>
<tr>
<td>DROP CLIENTSIDE ENCRYPTION KEYPAIR</td>
<td>21 - Client-side Encryption</td>
<td>Audits the dropping of a client-side encryption key pair.</td>
<td>No</td>
</tr>
<tr>
<td>PERSONAL DATA ACCESS</td>
<td>22 - Application Auditing</td>
<td>Audits audit log access to personal data.</td>
<td>No</td>
</tr>
<tr>
<td>PERSONAL DATA MODIFICATION</td>
<td>22 - Application Auditing</td>
<td>Audits audit log modification of personal data.</td>
<td>No</td>
</tr>
<tr>
<td>CONFIGURATION CHANGE</td>
<td>22 - Application Auditing</td>
<td>Audits audit log configuration change events.</td>
<td>No</td>
</tr>
<tr>
<td>SECURITY EVENT</td>
<td>22 - Application Auditing</td>
<td>Audits audit log security events.</td>
<td>No</td>
</tr>
<tr>
<td>DEBUGGER ATTACH PROCESS(^1)</td>
<td>23 - Procedure Debugging</td>
<td>Audits audit log debugger attachment process events.</td>
<td>No</td>
</tr>
<tr>
<td>DEBUGGER EXECUTION(^1)</td>
<td>23 - Procedure Debugging</td>
<td>Audits audit log debugger execution events.</td>
<td>No</td>
</tr>
<tr>
<td>ALTER CREDENTIAL</td>
<td>24 - Credential Management</td>
<td>Audits the use of ALTER CREDENTIAL commands.</td>
<td>No</td>
</tr>
<tr>
<td>CREATE CREDENTIAL</td>
<td>24 - Credential Management</td>
<td>Audits the use of CREATE CREDENTIAL commands.</td>
<td>No</td>
</tr>
<tr>
<td>DROP CREDENTIAL</td>
<td>24 - Credential Management</td>
<td>Audits the use of DROP CREDENTIAL commands.</td>
<td>No</td>
</tr>
<tr>
<td>IMPORT</td>
<td>25 - Data Import Export</td>
<td>Audits the import of data</td>
<td>No</td>
</tr>
<tr>
<td>IMPORT SCAN</td>
<td>25 - Data Import Export</td>
<td>Audits the scan of import data</td>
<td>No</td>
</tr>
<tr>
<td>EXPORT</td>
<td>25 - Data Import Export</td>
<td>Audits the export of data</td>
<td>No</td>
</tr>
</tbody>
</table>

\(^1\) This action can only be added to policies created on SYSTEMDB and does not support the FOR `<database_name>` option.
Permissions

To create an audit policy on SYSTEMDB or on a tenant, you must have the AUDIT ADMIN privilege.

To use the FOR <database_name> option, you need the DATABASE AUDIT ADMIN privilege granted on SYSTEMDB, not on the tenant.

Example

Create a new audit policy named priv_audit that audits successful granting and revoking of privileges and roles. The audit policy has the medium audit level CRITICAL.

This policy has to be explicitly enabled to cause the auditing of the audit policy.

```
CREATE AUDIT POLICY priv_audit AUDITING SUCCESSFUL GRANT PRIVILEGE, REVOKE PRIVILEGE, GRANT ROLE, REVOKE ROLE LEVEL CRITICAL;
```

Create a new audit policy named object_audit that audits the inserts into the existing table MY_SCHEMA.MY_TABLE. This policy must be explicitly enabled to cause the auditing of the audit policy. This policy is restricted to user FRED and uses the audit level INFO.

```
CREATE USER FRED PASSWORD <pwd>;  CREATE SCHEMA MY_SCHEMA OWNED BY system;
CREATE ROW TABLE MY_SCHEMA.MY_TABLE (first_col int);
GRANT INSERT ON MY_SCHEMA.MY_TABLE to FRED;
CREATE AUDIT POLICY OBJECT_AUDIT AUDITING SUCCESSFUL INSERT ON MY_SCHEMA.MY_TABLE FOR FRED LEVEL INFO;
```

Create a new audit policy named update_object_audit that audits the updates of the existing table MY_SCHEMA.MY_TABLE. This policy must be enabled explicitly to make the auditing of the audit policy occur. The auditing should be done for all users except the existing user TECH_ADMIN. This policy uses the audit level CRITICAL.

```
CREATE USER TECH_ADMIN PASSWORD <pwd>;
CREATE SCHEMA MY_SCHEMA OWNED BY system;
CREATE ROW TABLE MY_SCHEMA.MY_TABLE (first_col int);
GRANT UPDATE ON MY_SCHEMA.MY_TABLE to TECH_ADMIN;
CREATE AUDIT POLICY UPDATE_OBJECT_AUDIT AUDITING SUCCESSFUL UPDATE ON MY_SCHEMA.MY_TABLE EXCEPT FOR TECH_ADMIN LEVEL CRITICAL;
```

Create a new audit policy named SYSTEMDB_TEST that audits successful changes to an existing PSE. This policy is set on the SYSTEMDB and applies the tenant HA2.

```
CREATE AUDIT POLICY SYSTEMDB_TEST FOR HA2 AUDITING SUCCESSFUL ALTER PSE LEVEL CRITICAL;
```

Create an audit policy that tracks all tables created in the schema TEST_SCHEMA.

```
CREATE AUDIT POLICY MY_AUDIT_POLICY AUDITING ALL CREATE TABLE ON SCHEMA TEST_SCHEMA LEVEL INFO;
```
4.10.1.47 CREATE CERTIFICATE Statement (System Management)

Adds the specified certificate to the list of certificates.

Syntax

```
CREATE CERTIFICATE <certificate_name> FROM <certificate> [ COMMENT <comment> ]
```

Syntax Elements

**certificate_name**

If a `<certificate_name>` is not specified, a name is automatically generated from its ID in the format _SYS_CERTIFICATE_<id>. If migrating from an earlier release of SAP HANA database, a certificate name for existing certificates is automatically generated using the same format. The CERTIFICATE_NAME column has been added to the CERTIFICATES and PSE_CERTIFICATES System Views views.

```
<certificate_name> ::= <identifier>
```

**certificate**

Specifies the certificate in PEM-representation, which should be stored in the database.

```
<certificate> ::= <string literal>
```

**comment** Adds a description for the certificate.

```
<comment> ::= <string_literal>
```

Use this value to help you find a specific certificate in the list of certificates.
Description

The CREATE CERTIFICATE statement adds this certificate to the list of certificates, which can be assigned to a PSE store.

Information about certificates that are usable for assignment to PSE stores is stored in the CERTIFICATES system view.

Duplicate certificates are allowed as long as a certificate name is specified and unique.

Permissions

Only users with the system privilege CERTIFICATE ADMIN can create certificates.

Examples

Create a certificate with subject and issuer SAP AG and a validity between 2011 and 2285. The certificate is shortened and the missing 21 lines mentioned as '...'.

```
CREATE CERTIFICATE FROM '-----BEGIN CERTIFICATE-----
MIIEVjCCAz6gAwIBAgIJAKZmSWxYxVmGMA0GCSqGSIb3DQEBBQUAMHkxCzAJ
zn2Q+T5og6ozDIwqUYsegJl3W2gNznEj66Ku1SDDzR0POjCnfK5xLt1WE5KBAIav
1SSbStsw6rCRdg==
-----END CERTIFICATE-----' COMMENT 'Subject SAP AG Valid until 2285'
```

Create a certificate named SAP_CERT. The certificate is shortened and the missing 21 lines mentioned as '...'.

```
CREATE CERTIFICATE SAP_CERT FROM '-----BEGIN CERTIFICATE-----
MIIEVjCCAz6gAwIBAgIJAKZmSWxYxVmGMA0GCSqGSIb3DQEBBQUAMHkxCzA
zn2Q+T5og6ozDIwqUYsegJl3W2gNznEj66Ku1SDDzR0POjCnfK5xLt1WE5KBAIav
1SSbStsw6rCRdg==
-----END CERTIFICATE-----'
```

Related Information

CERTIFICATES System View [page 1516]
4.10.1.48 CREATE CLIENTSIDE ENCRYPTION COLUMN KEY
Statement (Encryption)

Creates a column encryption key (CEK) to be used in client-side encryption.

Syntax

```
CREATE CLIENTSIDE ENCRYPTION COLUMN KEY [ <schema_name>..<key_name> ]
    [ ALGORITHM <algorithm_name> ]
    { ENCRYPTED WITH KEYPAIR <keypair_name> | HEADER ONLY }
```

Syntax Elements

- **schema_name**
  Specifies the schema name for the CEK.
  
  `<schema_name> ::= <identifier>`

- **key_name**
  Specifies the name of the CEK.
  
  `<key_name> ::= <identifier>`

- **ALGORITHM algorithm_name**
  Specifies the algorithm to use for encryption of columns. Currently, only AES-256-CBC (default) and ARIA-256-CBC are supported.
  
  `<algorithm_name> ::= '<string>'`

- **ENCRIPTED WITH KEYPAIR keypair_name**
  Specifies the name of the client key pair (CKP) to use for encrypting the column key.
  
  `<keypair_name> ::= <identifier>`

- **HEADER ONLY**
  Creates an empty CEK that you can use for schema creation without requiring the creation of actual encryption keys.

Description

CEKs are symmetric keys used to encrypt sensitive data columns in the client drivers. Creating a CEK is required before any column in the database can be encrypted by using the client-side encryption feature.
Before creating an encrypted CEK, you must define a client key pair by using the CREATE CLIENTSIDE ENCRYPTION KEYPAIR statement.

The client can specify a key pair name (<keypair_name>) to encrypt the new CEK. The client driver collaborates with the SAP HANA server transparently to generate a new CEK encrypted with the specified public key (<keypair_name>). The new CEK is then stored in the SAP HANA database catalog.

Use HEADER ONLY to allow the client to proceed with database schema creation (with client-side encryption) without creating and specifying actual key information during creation. A CREATE TABLE statement may reference an empty CEK. An ALTER TABLE statement on an empty table may also reference an empty CEK. Any attempt to insert rows into the table fails until the CEK has been populated by an ALTER CLIENTSIDE ENCRYPTION COLUMN KEY statement.

Permissions

Specifying ENCRYPTED WITH KEYPAIR <keypair_name> requires the CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege. Creating an empty CEK (HEADER ONLY) requires the CREATE ANY or CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege.

Examples

The following statement creates an column encryption key hrapp_cek1 that uses the AES-256-CBC encryption algorithm, encrypted with key pair key_admin_ckp:

```
CREATE CLIENTSIDE ENCRYPTION COLUMN KEY myschema.hrapp_cek1 ALGORITHM 'AES-256-CBC' ENCRYPTED WITH KEYPAIR key_admin_ckp;
```

The following statement creates an empty column encryption key hrapp_cek2 that uses the AES-256-CBC encryption algorithm:

```
CREATE CLIENTSIDE ENCRYPTION COLUMN KEY hrapp_cek2 ALGORITHM 'AES-256-CBC'
HEADER ONLY;
```

Related Information

CLIENTSIDE_ENCRYPTION_COLUMN_KEYS System View [page 1517]
CLIENTSIDE_ENCRYPTION_KEYPAIRS System View [page 1519]
ALTER CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 435]
DROP CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 949]
CREATE CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 743]
DROP CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 950]
4.10.1.49 CREATE CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption)

Creates a client key pair (CKP).

Syntax

```
CREATE CLIENTSIDE ENCRYPTION KEYPAIR <keypair_name> [ ALGORITHM <algorithm_name> ]
```

Syntax Elements

- `keypair_name`
  Specifies the name of the CKP.

- `algorithm_name`
  Specifies the name of the algorithm the key uses to encrypt column encryption keys. Currently, only RSA-OAEP-2048 is supported, which is the default.

Description

The client-driver uses this statement to create a new CKP which the SAP HANA server stores in the catalog. Client key pairs are asymmetric keys used to distribute column encryption keys to clients. CKPs are generated by the client-driver, and (both the public and private key parts are) stored in the hdbkeystore on the client machine along with their name and UUID. The public key part of the CKP is stored in SAP HANA catalog.

On the SAP HANA server, CKPs are named database level objects. CKPs are not shared between different database systems. If a client accesses multiple databases that support client-side encryption, it will need a unique key pair for each one.

Permissions

The use of this statement requires the CREATE CLIENTSIDE ENCRYPTION KEYPAIR privilege.
**Examples**

The following statement creates a CKP called `user1_ckp` that uses the encryption algorithm `RSA-OAEP-2048`:

```
CREATE CLIENTSIDE ENCRYPTION KEYPAIR user1_ckp ALGORITHM 'RSA-OAEP-2048';
```

**Related Information**

- CLIENTSIDE_ENCRYPTION_COLUMN_KEYS System View [page 1517]
- CLIENTSIDE_ENCRYPTION_KEYPAIRS System View [page 1519]
- CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 741]
- ALTER CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 435]
- DROP CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 949]
- CREATE CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 743]
- DROP CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 950]
- hdbkeystore Utility
- ALTER CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 437]

**4.10.1.50  CREATE CREDENTIAL Statement (Access Control)**

Creates a component-specific or application-specific credential.

**Syntax**

```
CREATE CREDENTIAL FOR [USER <user_name>] COMPONENT <component_id>
    PURPOSE <purpose_def> TYPE <type_def> [USING <using_param>]
```

**Syntax Elements**

- `user_name`
  
  Specifies the user that can use the credential.

  `<user_name> ::= <unicode_name>`

  When this parameter is specified, the created credential is used for this database user only.
If USER `<user_name>` is not provided, then every user of the current instance uses the same credential when the following conditions are true:

- The user is using the specified component.
- The component needs access to the remote system specified by the PURPOSE parameter.

Only users with the CREDENTIAL ADMIN system privilege are allowed to alter credentials for other users.

**component_id**

Specifies an identifier for the component that uses the credential.

```
<component_id> ::= <string_literal>
```

**purpose_def**

Specifies the application-specific or component-specific purpose string.

```
<purpose_def> ::= <string_literal>
```

**type_def**

Specifies a connection mechanism. For example 'PASSWORD' or 'KERBEROS'.

```
<type_def> ::= <string_literal>
```

**using_param**

Specifies the connection parameter.

```
<using_param> ::= <string_literal>
```

This value uses a type specified by the `<type_def>` parameter. It is required if the type is 'PASSWORD'.

**Description**

The CREATE CREDENTIAL statement creates a component-specific or application-specific credential.

**Permissions**

This statement requires the CREDENTIAL ADMIN privilege.

**Examples**

Create a credential for all users that is used by the component INTERNAL_APP. This credential uses the password mechanism for the PURPOSE COMPANY_MASTER_MACHINE.

```
CREATE CREDENTIAL FOR COMPONENT 'INTERNAL_APP' PURPOSE 'COMPANY_MASTER_MACHINE'
    TYPE 'PASSWORD' USING 'PASSWORD_9876';
```
Create a credential for all users that is used by the component SAPHANAFEDERATION. This credential uses Kerberos for authentication.

```
CREATE CREDENTIAL FOR COMPONENT 'SAPHANAFEDERATION' PURPOSE 'H1' TYPE 'KERBEROS';
```

Related Information

CREDENTIALS System View [page 1524]

4.10.1.51 CREATE DATABASE Statement (Tenant Database Management)

Creates a tenant database.

Syntax

```
CREATE DATABASE <database_name> [ AT [ LOCATION ] '<hostname>[:<port_number> ] ' ]
[ ADD '<servicetype>' [ AT [ LOCATION ] '<hostname>[:<port_number> ] ' ] ]
[ { AS REPLICA OF [ <source_database_name> ] AT [ LOCATION ]
'<hostname>[:<port_number> ] ' ] |
{ KEY MANAGEMENT CONFIGURATION <configuration_id> } ] | SYSTEM USER
PASSWORD <password> ] ]
[ OS USER '<username>' OS GROUP '<groupname>' ]
[ NO START ]
[ <restart_mode> RESTART ]
```

Syntax Elements

- **database_name**
  Specifies a unique name for the tenant database.

  ```
  <database_name> ::= <identifier>
  ```

- **hostname**
  Specifies the name of the host to create the tenant database on.

- **port_number**
  Specifies the port number for the host.

- **ADD servicetype**
  Adds a service type to the tenant database.

- **AS REPLICA OF**
  Creates a replica of another database.

- **KEY MANAGEMENT CONFIGURATION**
  Configures the key management for the database.

- **SYSTEM USER PASSWORD**
  Specifies the password for the system user.

- **OS USER OS GROUP**
  Specifies the operating system user and group for the database.

- **NO START**
  Specifies that the database should not start automatically.

- **<restart_mode> RESTART**
  Specifies the restart mode for the database.
Specifies a service type for the tenant database.

```
<servicetype> ::= <identifier>
```

**AS REPLICA OF** [source_database_name]

Specifies the unique name of the source database to be replicated.

```
<source_database_name> ::= <identifier>
```

**KEY MANAGEMENT CONFIGURATION** configuration_id

Specifies the key management configuration, not the user ID to use on the target system. The configuration ID must exist on the source system and is only necessary if the source system uses local secure store (LSS) with an active key management system (KMS). The user ID specified within the configuration must have been granted the ENCRYPTION ROOT KEY ADMIN system privilege. The `<configuration_id>` can be obtained from the `key_management_configuration` view if you have the ENCRYPTION ROOT KEY ADMIN system privileges. For more information, see Create a Key Management Configuration in the SAP HANA Administration Guide for SAP HANA Platform.

**SYSTEM USER PASSWORD** password

Specifies a password for the tenant database.

**OS USER** username

The operating system user under which the tenant is executed.

```
<username> ::= <identifier>
```

The user must exist already. If this value is not specified, then the `sidadm` user is used.

**OS GROUP** groupname

Specifies the operating system group under which the tenant is executed.

```
<groupname> ::= <identifier>
```

The group must exist already. If this value is not specified, then the `sapsys` user is used.

**restart_mode** RESTART

Specifies the restart mode for the tenant database after a system restart.

```
<restart_mode> ::= DEFAULT | NO
```

**DEFAULT** RESTART restores the previous state of the tenant database before the system restart. For example, if the tenant database was not started before the system restart, then it is not started after the system restart. This is the default behavior.

**NO** RESTART specifies to not restart the tenant database after a system restart.

**Description**

You can create multiple databases.

With the **AS REPLICA OF** clause, you can create the tenant database as a replica of an existing tenant database.
Permissions

The use of this statement requires the DATABASE ADMIN privilege.

Example

Create a tenant database with automatic host placement and automatic port assignment.

```
CREATE DATABASE DB0 SYSTEM USER PASSWORD Manager1;
```

Create a tenant database on a specific host accessible via a specific SQL port (30148). The SQL port number is the internal communication port number plus 1.

```
CREATE DATABASE DB1 AT LOCATION 'HOST_A:30147' SYSTEM USER PASSWORD Manager1;
```

Create a tenant database on a specific host, adding an additional slave index server on a specific second host and an auxiliary script server on an automatically selected host.

```
CREATE DATABASE DB2 AT LOCATION 'HOST_A' ADD 'indexserver' AT 'HOST_B' ADD 'scriptserver' SYSTEM USER PASSWORD Manager1;
```

Related Information

System Limitations [page 1208]
Create and Manage a Key Management Configuration
KEY_MANAGEMENT_CONFIGURATIONS System View [page 1592]

4.10.1.52 CREATE FULLTEXT INDEX Statement (Data Definition)

Creates an explicit fulltext index on the specified table column.

Syntax

```
CREATE FULLTEXT INDEX <index_name> ON <table_name> ( <column_name> ) [ <fulltext_parameter_list> ]
```
Syntax Elements

**index_name**

Specifies the name of the fulltext index to be created, with optional schema name.

```
<index_name> ::= [ <schema_name>.]<identifier>
<schema_name> ::= <unicode_name>
```

**fulltext_parameter_list**

Specifies the fulltext parameters.

```
<fulltext_parameter_list> ::= <fulltext_parameter> [, <fulltext_parameter> [, … ] ]
<fulltext_parameter> ::=  LANGUAGE COLUMN <column_name>
  LANGUAGE DETECTION ( <string_literal_list> )
  MIME TYPE COLUMN <column_name>
  <change_tracking_elem>
  FUZZY SEARCH INDEX { ON | OFF }
  PHRASE INDEX RATIO <index_ratio>
  CONFIGURATION <string_literal>
  SEARCH ONLY { ON | OFF }
  FAST PREPROCESS { ON | OFF }
  TEXT ANALYSIS { ON | OFF }
  MIME TYPE <string_literal>
  TOKEN SEPARATORS <string_literal>
  TEXT MINING { ON | OFF }
  TEXT MINING CONFIGURATION OVERLAY <string_literal>
<column_name> ::= <identifier>
```

**LANGUAGE COLUMN column_name**

Defines the column where the language of a document is specified.

**LANGUAGE DETECTION ( string_literal_list )**

Specifies the set of languages to be considered during language detection.

**MIME TYPE COLUMN column_name**

Defines the column where the mime-type of a document is specified.

**change_tracking_elem**

Specifies the type of index to be created.

```
<change_tracking_elem> ::=  SYNC[HRONOUS]
                         | ASYNC[HRONOUS] [FLUSH [QUEUE] <flush_queue_elem>]
```

**SYNC[HRONOUS]**

Creates a synchronous fulltext index.

**ASYNC[HRONOUS]**

Creates an asynchronous fulltext index.

**flush_queue_elem**

Specifies when to update the fulltext index if an asynchronous index is used.

```
FLUSH [QUEUE] <flush_queue_elem>
```
flush_queue_elem ::= EVERY <integer_literal> MINUTES  | AFTER <integer_literal> DOCUMENTS  
| EVERY <integer_literal> MINUTES OR AFTER <integer_literal> DOCUMENTS

When DOCUMENTS is specified, the fulltext index is updated after the specified number of changes to the table, including updates and deletes.

FUZZY SEARCH INDEX { ON | OFF }

Specifies whether a fuzzy search index should be used.

PHRASE INDEX RATIO <index_ratio>

Specifies the percentage of the phrase index. Value must be between 0.0 and 1.0

PHRASE INDEX RATIO <index_ratio>  
<index_ratio> ::= <exact_numeric_literal>

CONFIGURATION string_literal

Specifies the name of a standard text analysis configuration, or the fully-qualified name of a custom text analysis configuration stored in the HANA Repository, created with the HANA Deployment Infrastructure, or created with the TEXT_CONFIGURATION_CREATE stored procedure.

If the configuration was created with HANA DI, then the HANA DI container must be the same as the schema for the fulltext index. If the configuration was created with TEXT_CONFIGURATION_CREATE, then the configuration must have been created with the same schema name as the schema for the fulltext index.

SEARCH ONLY { ON | OFF }

Controls whether the original document should be stored (OFF) or only the search relevant tokens. If set to ON, then unsearchable parts of the document are removed.

FAST PREPROCESS { ON | OFF }

Controls whether fast preprocessing is used. If set to ON, then fast preprocessing is used and linguistic searches are not possible.

TEXT ANALYSIS { ON | OFF }

Enables text analysis capabilities on the indexed column. Text analysis can extract entities such as persons, products, or places from documents, which are stored in a new table.

MIME TYPE string_literal

Specifies the default mime type used for preprocessing. The value must be a valid mime type. For example, 'cf M_TEXT_ANALYSIS_MIME_TYPES'.

TOKEN SEPARATORS string_literal

Specifies a set of characters used for token separation. Only ASCII characters are considered.

TEXT MINING { ON | OFF }

Controls whether to enable text mining. If set to ON, then text mining capabilities will be initialized on the indexed column when the fulltext index is created. Text mining provides functionality that can compare documents by examining the terms used within them.

TEXT MINING CONFIGURATION string_literal

Specifies the name of a standard text mining configuration, or the fully-qualified name of a custom text mining configuration stored in the HANA Repository, created with the HANA Deployment Infrastructure, or created with the TEXT_CONFIGURATION_CREATE stored procedure. If the
configuration was created with HANA DI, then the HANA DI container must be the same as the schema for the fulltext index. If the configuration was created with TEXT_CONFIGURATION_CREATE, then the configuration must have been created with the same schema name as the schema for the fulltext index.

**TEXT MINING CONFIGURATION OVERLAY string_literal**

Specifies literal text mining configuration data that should override the text mining configuration file. The format is the same as in the configuration file, typically a small subset of parameters. This allows parameter experimentation without requiring creation of configuration files for each case.

**Description**

A fulltext index is supported for the following data types:

- VARCHAR, NVARCHAR types with CS_STRING type
- VARBINARY type
- BLOB, CLOB, NCLOB types

**Example**

Create table A.

```
CREATE COLUMN TABLE A (A VARCHAR(10) PRIMARY KEY, B VARCHAR(10));
```

Create a synchronous fulltext index named i. Index i is on column A of table A, where the fuzzy search index is not used and the set of languages for language detection consists of EN, DE, and KR.

```
CREATE FULLTEXT INDEX i ON A(A) FUZZY SEARCH INDEX OFF SYNC LANGUAGE DETECTION ('EN','DE','KR');
```

**Related Information**

SAP HANA Text Mining Developer Guide
4.10.1.53 CREATE FUNCTION Statement (Procedural)

Creates a user-defined function.

Syntax

```sql
CREATE [ OR REPLACE ] FUNCTION <function_name> [ ( <input_parameter_clause> ) ] RETURNS <return_type> [ LANGUAGE <lang> ] [ SQL SECURITY { DEFINER | INVOKER } ] [ DEFAULT SCHEMA <default_schema_name> ] [ <variable_cache_clause> ] [ DETERMINISTIC ] [ WITH ENCRYPTION ] AS { BEGIN <statement_body> END [ <cache_clause> ] | HEADER ONLY }
```

Syntax Elements

**function_name**

Specifies the function name, and optionally, a schema name.

```sql
<function_name> ::= [ <schema_name>.]identifier  
<schema_name> ::= <unicode_name>
```

**input_parameter_clause**

Specifies the input parameters for the function.

```sql
<input_parameter_clause> ::= <parameter> [,...]  
<parameter> ::= [ IN ] <param_name> <data_type>  
<param_name> ::= ...  
<data_type> ::= <sql_type> [ ARRAY ]  
|<table_type>  
|<table_type_definition>  
|<any_table_type>
```

Scalar user-defined functions support primitive SQL types, as well as table parameters (actual physical tables), views, and table variables. Table user-defined functions support table types as input.

**sql_type [ ARRAY ]**

Specifies the SQL type. Use the optional ARRAY keyword with `<sql_type>` if the parameter is an array.

```sql
<sql_type> ::=  
DATE  
TIME  
TIMESTAMP
```
Scalar user-defined functions do not support ALPHANUM, VARBINARY, CLOB, NCLOB, or BLOB types.

**table_type**

Specifies a table type that was previously defined with the CREATE TYPE statement.

```
<table_type> ::= [ <schema_name>.]<identifier>  
<schema_name> ::= <unicode_name>
```

**table_type_definition**

Specifies a table type that is implicitly defined within the signature.

```
<table_type_definition> ::= TABLE (<column_list_definition>)  
<column_list_definition> ::= <column_elem>[, <column_elem> [,…] ]  
<column_elem> ::= <column_name> <data_type>  
<column_name> ::= <identifier>
```

**any_table_type**

Defines the table type at DDL time or query compilation time.

```
<any_table_type> ::= TABLE(...)  
```

The syntax of `<any_table_type>` requires you to explicitly specify `TABLE(...)` (for example, `CREATE FUNCTION anyfunc_in (IN itab TABLE(...), outtab TABLE(...) ) AS BEGIN...`).

**RETURNS return_type**

Specifies the return type.

```
<return_type> ::= { <return_parameter_list> | <return_table_type> }  
```

Scalar functions must return scalar values specified in `<return_parameter_list>`. Table functions must return a table whose type is defined by `<return_table_type>.

**return_parameter_list**

Specifies the output parameters. Use the optional ARRAY keyword for `<sql_type>` if the return parameter is an array.

```
$return_parameter_list ::= <return_parameter>[, <return_parameter> [,…] ]  
$return_parameter ::= <parameter_name> <sql_type> [ ARRAY ]
```

**return_table_type**
Specifies the structure of the returned table data.

\[
\text{<return_table_type>} ::= \text{TABLE} \left( \text{<ret_column_list>} \right)
\]

\text{ret_column_list}

Specifies the list of columns returned from the function.

\[
\text{<ret_column_list>} ::= \text{<ret_column_elem>}[, \text{<ret_column_elem>} [,…]} \]

\text{ret_column_elem}

Specifies the name of the column element with its associated data type.

\[
\text{<ret_column_elem>} ::= \text{<column_name>} \text{<sql_type>}
\]

\text{LANGUAGE} \text{lang}

Specifies the programming language used in the function.

\[
\text{LANGUAGE} \text{<lang>}
\]

\[
\text{<lang>} ::= \text{SQLSCRIPT}
\]

You can only use SQLScript functions.

\text{SQL SECURITY} \{ \text{DEFINER} | \text{INVOKER} \}

Specifies the security mode of the function.

\text{DEFINER}

Specifies that the function is executed with the privileges of the definer of the function. DEFINER is the default for table user-defined functions.

\text{INVOKER}

Specifies that the function is executed with the privileges of the invoker of the function. Invoker is the default for scalar user-defined functions.

\text{default_schema_name}

\text{minute_value}

Specifies the result cache retention period.

\[
\text{<minute_value>} ::= \text{<unsigned_integer>}
\]

Only stale data access that does not exceed the specified RETENTION period is allowed. For outdated data that exceeds the RETENTION period, internally the result cache is refreshed and up-to-date data is returned.

Specifies the schema for unqualified objects in the function body.

\[
\text{<default_schema_name>} ::= \text{<identifier>}
\]

If \text{<default_schema_name>} is not specified, then the \text{<current_schema>} of the session is used.

\text{DETERMINISTIC}

DETERMINISTIC applies to scalar-functions. Signifies that the SQLScript function returns the same result any time it is called with the same input parameters.

\text{cache_clause}
Caches the result of a table function with an optional projection list and filter conditions.

Using a static result cache may result in stale data, and specifying a retention period ensures that data does not exceed the specified RETENTION period. Data that exceeds the retention period is refreshed and up-to-date data is returned.

```sql
<cache_clause> ::= WITH [ STATIC ] CACHE
[ RETENTION <minute_value> ]
[ OF <projection_list> ]
[ FILTER <filter_condition> ]
[ <location_clause> ]
```

**minute_value**

Specifies the result cache retention period.

```sql
<minute_value> ::= <unsigned_integer>
```

Only stale data access that does not exceed the specified RETENTION period is allowed. For outdated data that exceeds the RETENTION period, internally the result cache is refreshed and up-to-date data is returned.

**projection_list**

Specifies a result cached projection list to reduce the amount of cached data.

```sql
<projection_list> ::= <projection_name> [, <projection_name> [,...] ]
<projection_name> ::= <column_name>    | <aggr_type>(<column_name>)
<aggr_type> ::= { SUM | MIN | MAX | COUNT | AVG }
```

If a column that is not part of the projection list is requested or included in the WHERE clause, then the result cache cannot be exploited. In addition, you can direct an aggregation type of a specific column. If the column is specified, then the result cache includes aggregated results of that column and returns aggregated results only.

**filter_condition**

Specifies a filter condition to reduce the amount of cached data.

```sql
<filter_condition> ::=<condition> OR <condition>
| <condition> AND <condition>
| NOT <condition>
| ( <condition> )
| <predicate>

<predicate> ::=<comparison_predicate>
| <range_predicate>
| <in_predicate>
| <like_predicate>
| <null_predicate>

<comparison_predicate> ::=<table_expression> { = | != | < | > | >= | <= } [ ANY | SOME | ALL ]
( <table_expression_list> )

<range_predicate> ::=<table_expression> [ NOT ] BETWEEN <table_expression> AND
<table_expression>

<in_predicate> ::=
Only filtered results are cached. Predicates with subqueries are not supported. All forms of non-parameterized query filters are supported. For parameterized query filters, only conjunctive forms are supported.

**location_clause**

Specifies the location for the result cache entry.

```
<locationClause> ::= AT [LOCATION] '<hostname>:<port_number>', ...
```

By default, when you execute a query on a static result cached view, the optimizer determines the result cache location. In some cases, the same result cache entry is created at multiple index servers.

**variable_cache_clause**

Caches variable values during execution time so that they do not need to be re-resolved as the operations in <statement_body> are executed. Caching a variable removes the risk that its value will change each time it is resolved.

```
<variable_cache_clause> ::= VARIABLE CACHE ON {
  <variable_entry_with_mode_list> [ <variable_entry_without_mode_list> ]
  <variable_entry_with_mode_list> ::= <variable_entry_with_mode> [, ...
  <variable_entry_with_mode> ::= <variable_entry enable_mode> ...
  <variable_entry> ::= { <variable_name> | <variable_name_clustered_list> }
  <variable_name_clustered_list> ::= ( <variable_name> [, <variable_name> ] )
  <variable_name enable_mode> ::= { ENABLE | DISABLE }
  <variable_entry_without_mode_list> ::= <variable_entry> [, <variable_entry> ]
```

**WITH ENCRYPTION**

Encrypts the definition of the function. You cannot decrypt the definition later.

**statement_body**

Specifies the main body of the table functions and scalar functions.

```
<statement_body> ::= [ <func_using_list> ] [ <func_decl_list> ] [ <func_handler_list> ] <func_stmt_list>
```

Since the function is flagged as read-only, neither DDL or DML statements (INSERT, UPDATE, and DELETE) are allowed in the function body. Scalar functions support table-typed input variables and table cell assignments. Other table operations are not supported.

For more information on <func_using_list>, <func_decl_list>, <func_handler_list>, <func_stmt_list>, see the corresponding <proc_using_list>, <proc_decl_list>, <proc_handler_list>, and <proc_stmt_list> clause in the CREATE PROCEDURE Statement (Procedure).
**Description**

The CREATE FUNCTION statement creates read-only functions that are free of side effects. Neither DDL or DML statements (INSERT, UPDATE, and DELETE) are allowed in the function body. Also, other functions or procedures selected/called from the body of the function must be read only.

There are two kinds of user-defined function (UDF): table and scalar. They differ by input/output parameter, supported functionality in the body, and the way that they are consumed in SQL statements.

User-defined functions are read-only functions that are free of side effects: DDL or DML statements are not allowed within the function body.

When you specify HEADER ONLY, only the properties of the function are created along with the OID and no function dependencies exist. Once the headers are replaced with full function definitions, dependencies are created while the function OID remains the same. Dependencies appear in the OBJECT_DEPENDENCIES system view. HEADER ONLY is useful when you need to create dependent functions.

**Notes on CREATE OR REPLACE behavior:**

- The behavior of CREATE OR REPLACE depends on the existence of a pre-defined function: If there is no existing function, then it behaves like CREATE FUNCTION; otherwise, it behaves like ALTER FUNCTION.
- You must explicitly set all parameters where the desired setting is different from the default (for example, settings for procedure language, default schema, whether the procedure is deterministic or encrypted, sequential execution options, and so on). Everything not specified in a CREATE OR REPLACE FUNCTION statement is reset to its default.
- You cannot use CREATE OR REPLACE FUNCTION to force a recompile of an existing function.
- If you specify the security mode or route options, then the behavior depends on the existence of a pre-defined function: If there is no existing function, then the operation succeeds (the function is created). If the function exists and the security mode is different, then an error is returned.
- CREATE OR REPLACE does not change any DDL settings for the function, such as grant privileges, dependent objects, and so on.

See the SAP HANA SQLScript Reference guide for more information about creating and modifying user-defined functions.

**Examples**

Create a table function called **Scale**.

```sql
CREATE FUNCTION Scale (val INT)  
RETURNS TABLE (a INT, b INT) LANGUAGE SQLSCRIPT AS 
BEGIN
  RETURN SELECT a, :val * b AS  b FROM mytab;
END;
```

Use the **func_add_mul** function like a table. For example:

```sql
SELECT * FROM Scale(10);
SELECT * FROM Scale(10) AS a, Scale(10) AS b WHERE a.a = b.a;
```
Create a scalar function called `func_add_mul`.

```sql
CREATE FUNCTION func_add_mul(x Double, y Double) RETURNS result_add Double, result_mul Double
LANGUAGE SQLSCRIPT READS SQL DATA AS
BEGIN
    result_add := :x + :y;
    result_mul := :x * :y;
END;
```

Use the `func_add_mul` function like a built-in function. For example:

```sql
CREATE ROW TABLE TAB (a Double, b Double);
INSERT INTO TAB VALUES (1.0, 2.0);
INSERT INTO TAB VALUES (3.0, 4.0);
SELECT a, b, func_add_mul(a, b).result_add AS ADD, func_add_mul(a, b).result_mul
AS MUL FROM TAB ORDER BY a;
```

The SELECT statement returns the following results:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>ADD</th>
<th>MUL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>7</td>
<td>12</td>
</tr>
</tbody>
</table>

Create a function called `func_mul` that is assigned to a scalar variable in the `func_mul_wrapper` function.

```sql
CREATE FUNCTION func_mul(input1 INT) RETURNS output1 INT LANGUAGE SQLSCRIPT AS
BEGIN
    output1 := :input1 * :input1;
END;
CREATE FUNCTION func_mul_wrapper(input1 INT) RETURNS output1 INT LANGUAGE SQLSCRIPT AS
BEGIN
    output1 := func_mul(:input1);
END;
SELECT func_mul_wrapper(2) as RESULT FROM DUMMY;
```

The SELECT statement returns 4.

Create a function called `FuncHeader` as HEADER ONLY. Later, you use ALTER FUNCTION to replace the header with the full function definitions.

```sql
CREATE FUNCTION FuncHeader (input1 integer) RETURNS output1 integer AS HEADER ONLY;
ALTER FUNCTION FuncHeader (input1 integer) RETURNS output1 integer
AS BEGIN
    output1 := :input1 * :input1;
END;
```

Create a table function called `FUNC` with a static result cache.

```sql
CREATE FUNCTION FUNC RETURNS TABLE (COL1 INT, COL2 INT) AS BEGIN
    RETURN SELECT * FROM TAB;
END
```

---

SAP HANA SQL Reference Guide for SAP HANA Platform

SQL Reference
WITH CACHE RETENTION 10;

Related Information

ALTER FUNCTION
CREATE FUNCTION
Table Variable Type Definition
M_SQLSCRIPT_VARIABLE_CACHE System view [page 2142]
CREATE TYPE Statement (Procedural) [page 892]
CREATE PROCEDURE Statement (Procedural) [page 776]
Expressions [page 56]
Predicates [page 61]
OBJECT_DEPENDENCIES System View [page 1604]
CREATE VIRTUAL FUNCTION Statement (Procedural) [page 917]

4.10.1.54  CREATE FUZZY SEARCH INDEX Statement (Data Definition)

Create a fulltext index on the specified table column. This statement is an alternative to using the CREATE FULLTEXT INDEX Statement.

Syntax

```
CREATE FUZZY SEARCH INDEX <index_name> ON <table_name>      ( <column_name> ) SEARCH MODE TEXT
```

Description

The fulltext index that is created by this statement has a fixed set of fulltext index parameters that cannot be changed by the user as follows:

<table>
<thead>
<tr>
<th>Fulltext Index Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAST PREPROCESS</td>
<td>ON</td>
</tr>
<tr>
<td>FUZZY SEARCH INDEX</td>
<td>ON</td>
</tr>
<tr>
<td>SEARCH ONLY</td>
<td>OFF</td>
</tr>
<tr>
<td>ASYNC</td>
<td></td>
</tr>
</tbody>
</table>

The DROP INDEX and RENAME INDEX statements can be used to managed fuzzy search indexes.
Permissions

Requires INDEX object privilege on the affected table to create an index.

Example

This example creates a fuzzy search index named fzyidx1_T1 on column a1 in table T1.

```sql
CREATE FUZZY SEARCH INDEX fzy_idx1 ON T1 (a1) SEARCH MODE TEXT;
```

4.10.1.55 CREATE GRAPH WORKSPACE Statement (Data Definition)

Creates a graph workspace.

Syntax

```sql
CREATE GRAPH WORKSPACE <workspace_name>
    EDGE TABLE <edge_table_name>
        SOURCE COLUMN <source_column_name>
        TARGET COLUMN <target_column_name>
        KEY COLUMN <edge_key_column_name>
    VERTEX TABLE <vertex_table_name>
        KEY COLUMN <vertex_key_column_name>
```

Syntax Elements

- **workspace_name**
  Specifies a name for the workspace.
- **edge_table_name**
  Specifies the name of the table containing the relationships between vertices (edge table).
  ```sql
  <edge_table_name> ::= [<schema_name>.]<identifier>
  ```
- **source_column_name**
  Specifies the source column for an edge in the graph.
  ```sql
  <source_column_name> ::= <identifier>
  ```
- **target_column_name**
Specifies the target column for an edge in the graph.

```
<target_column_name> ::= <identifier>
```

**edge_key_column_name**

Specifies unique IDs for the edges in the graph.

```
<edge_key_column_name> ::= <identifier>
```

**vertex_table_name**

Specifies the name of the table containing the vertices (members) of the graph.

```
<vertex_table_name> ::= [<schema_name>.]<identifier>
```

**vertex_key_column_name**

Specifies unique IDs for the vertices in the graph.

```
<vertex_key_column_name> ::= <identifier>
```

### Description

A graph is modeled as a set of vertices and a set of edges. An edge connects two vertices: one vertex is denoted as source and the other as target. There might be two or more edges connecting the same two vertices. The set of vertices is stored in a vertex table and the set of edges is stored in an edge table. One of the vertex columns needs to uniquely identify vertices. Such column is also referred to as vertex key. Similarly, one of the edge columns needs to uniquely identify edges and is referred to as edge key. The edge table contains two additional columns referencing the key column of the vertex table. One of them identifies the source vertex and the other identifies the target vertex.

Key columns of the edge and vertex columns must be UNIQUE and NOT NULL. The target and source columns must be NOT NULL and of the same type as the vertex key column. The data types VARCHAR, NVARCHAR and INT can be used for key columns.

Extensive documentation on using graphs is found in the SAP HANA Graph Reference.

### Example

Create the graph workspace "FAMILY_TREE.GENEALOGY" by defining the vertices and edges for the family tree.

```sql
CREATE SCHEMA "FAMILY_TREE";
CREATE COLUMN TABLE "FAMILY_TREE"."VERTICES" ("KEY" VARCHAR(1024) NOT NULL UNIQUE);
CREATE COLUMN TABLE "FAMILY_TREE"."EDGES" ("SOURCE" VARCHAR(1024) NOT NULL, "TARGET" VARCHAR(1024) NOT NULL, "ID" INT NOT NULL UNIQUE);
CREATE GRAPH WORKSPACE FAMILY_TREE.GENEALOGY
  EDGE TABLE FAMILY_TREE.EDGES
    SOURCE COLUMN "SOURCE"
    TARGET COLUMN "TARGET"
    KEY COLUMN ID
```
Related Information

4.10.1.56 CREATE INDEX Statement (Data Definition)

Creates an index on a table column.

Syntax

```sql
CREATE [ <index_type> ] INDEX <index_name> ON <table_name>
( <column_name_order_entry>[, <column_name_order_entry> [,...]])
[ <global_index_order> ]
[ <fillfactor> ]
[ NOWAIT ]
[ ONLINE ]
[ <load_unit> ]
```

Syntax Elements

**index_type**

Defines the type of index to create.

```sql
<index_type> ::= UNIQUE | [ UNIQUE ] <rs_tree_type_index> | [ UNIQUE ] <cs_inverted_type_index>
```

When the index type (`<rs_tree_type_index>` or `<cs_inverted_type_index>`) is omitted, the database server chooses the appropriate type by considering the table type and the column data type.

'rs_' and 'cs_' at the beginning of the specification types indicate which type of table the specification corresponds to (row store and column store, respectively). If you mistakenly specify a row store index type and the table is a column store table, then the specification is ignored and the index is created as INVERTED VALUE (default). If you mistakenly specify a column store index type and the table is a row store table, an error is returned.

**UNIQUE**

Specifies that the index must have unique values. The optional `<rs_tree_type_index>` specification is only applicable to row store tables.
A composite unique key enables the specification of multiple columns as a unique key. With a unique constraint, multiple rows cannot have the same value in the same column.

A UNIQUE column can contain multiple NULL values.

**rs_tree_type_index**

Specifies the tree type for the index. Tree type specification is only applicable to row store tables.

```plaintext
<rs_tree_type_index> ::= { BTREE | CPBTREE }
```

**CPBTREE**

Specifies a CPB+-tree index for row store tables. CPB+-tree stands for Compressed Prefix B+-Tree, which is based on pkB-tree. CPB+-tree is a very small index because it uses ‘partial key’ that is only part of full key in index nodes. CPB+-tree shows better performance than B+-Tree for larger keys. Specify this type of tree for the following scenarios:

- for character string types
- for binary string types
- for decimal types
- when the constraint is a composite key
- when the constraint is a non-unique constraint

**BTREE**

Specifies a B+-tree index. B+-tree indexes maintain sorted data, which performs efficient insertion, deletion, and search of records. Specify this type of tree for scenarios not described for CPBTREE.

**cs_inverted_type_index**

Specifies the inverted index type for column store tables. Inverted index types are only applicable to column store tables.

```plaintext
<cs_inverted_type_index> ::= INVERTED { HASH | VALUE | INDIVIDUAL }
```

**INVERTED HASH**

INVERTED HASH should not be used as a composite type in cases where range queries or similar queries on the composite keys are a significant part of the workload. In these cases, use INVERTED VALUE instead.

**i Note**

Non-unique INVERTED HASH indexes are deprecated. Only UNIQUE INVERTED HASH indexes are supported.

**INVERTED VALUE**

INVERTED VALUE is the default index type for column store tables.

**INVERTED INDIVIDUAL**

An INVERTED INDIVIDUAL index is a lightweight index type for column store tables with reduced memory footprint. The name INVERTED INDIVIDUAL reflects that internally only inverted indexes on individual columns are created (that is, no concat attributes are created). This type of index is only available for multi-column unique constraints, which may be defined as secondary unique index or primary key.
**index_name**

Specifies the name of the index to be created, with optional schema name.

```
<index_name> ::= [ <schema_name>.]<identifier>
```

**column_name_order_entry** Specifies the columns to be used in the index, with an optional ordering.

```
<column_name_order_entry> ::= <column_name> [ <column_order> ]
```

**column_order** Specifies whether the column index should be created in ascending or descending order. The default ordering is ASC.

```
<column_order> ::= { ASC | DESC }
```

**global_index_order** Specifies the index ordering for all columns in the index.

```
<global_index_order> ::= { ASC | DESC }
```

<column_order> and <global_index_order> cannot be used when <global_index_order> is used.

**fillfactor** Specifies how each node of a new index is filled. It is a percentage value of integer from 50 to 100, and the default value is 90.

```
<fillfactor> ::= FILLFACTOR <unsigned_integer>
```

**NOWAIT**

Specifies that the CREATE INDEX statement returns an error immediately in case a table lock cannot be acquired.

**ONLINE**

For column store tables, the ONLINE keyword allows the creation of the index without serializing with concurrent DML operations. This is only supported for non-replicated, non-history tables. Creating an index in a supported case results in an operation that acquires a shared table lock. In the unsupported cases, or when the ONLINE keyword is omitted, an exclusive table lock is only taken for a short time at the beginning and end of the CREATE INDEX operation; during the rest of the CREATE INDEX operation, shared table lock is taken, which allows concurrent read and DML access operations but blocks concurrent DDL operations.

For row store tables, you can specify ONLINE to not place an exclusive lock on the table and enable read access during index creation; if ONLINE is not specified neither concurrent read nor DML operations are allowed during index creation.

Note that the ONLINE keyword affects SQL table locks, but not other internal locks. Therefore a DML write or read operation could be blocked for some time even if the ONLINE keyword is specified. This blocking time is typically shorter than the complete index creation time, but especially in the case of a single column index creation it could even last for the complete index creation time.

**load_unit**

Specifies the load unit for the index. This clause is supported only for multi-column INVERTED VALUE indexes on column store tables.

```
<load_unit> ::= { COLUMN | PAGE } LOADABLE
```

**COLUMN LOADABLE** The query results are loaded by column. This is the default for column store tables.
**PAGE LOADABLE** The query results are loaded by page.

Column store inverted indexes can be as large as the column itself. The `<load_unit>` setting of an index impacts the memory footprint of the index. For example, a PAGE LOADABLE index consumes much less memory, but may degrade performance slightly. This may be suitable for an index that is not frequently used. When performance is the priority, you may want to set the index to COLUMN LOADABLE, assuming the memory overhead is acceptable.

The load unit for an index is noted in the LOAD_UNIT column of the INDEXES system view.

You cannot use the `<load_unit>` clause to create an INVERTED HASH index since it supports only dictionary-based inverted hash keys. If you attempt to create it, the following type of error is returned:

```
*7: feature not supported: Dictionary based inverted hash index with load unit
hint: line 1 col 44 (at pos 43) SQLSTATE: HY000
```

**Description**

When column data types are character string types, binary string types, decimal types, or when the constraint is a composite key, or a non-unique constraint, the default index type is CPBTREE. In other cases BTREE is used. If neither BTREE nor CPBTREE keyword is specified, then the SAP HANA database chooses the appropriate index type.

For row store tables, when column data types are character string types, binary string types, decimal types, or when the constraint is a composite key, or a non-unique constraint, the default index type is CPBTREE. In other cases BTREE is used. If neither BTREE nor CPBTREE keyword is specified, then the SAP HANA database chooses the appropriate index type.

For column store tables, the default index type is always INVERTED VALUE. Other index-types, such as INVERTED HASH or INVERTED INDIVIDUAL, have to be explicitly specified.

To determine the maximum index key size, check the MAXIMUM_SIZE_OF_KEY_IN_INDEX value in the M_SYSTEM_LIMITS system view.

**Example**

### Row store index example

Create a row store table `t`, and then add the index `idx` on column `b` of table `t` with ascending order.

```sql
CREATE ROW TABLE t (a INT, b NVARCHAR(10), c NVARCHAR(20));
CREATE INDEX idx ON t(b);
```

Create a CPBTREE index `idx1` on column `a` of table `t` with ascending order and column `b` with descending order.

```sql
CREATE CPBTREE INDEX idx1 ON t(a, b DESC);
```

Create a CPBTREE index `idx2` on column `a` and `c` of table `t` with descending order.

```sql
CREATE INDEX idx2 ON t(a, c) DESC;
```
Create a UNIQUE CPBTREE index \texttt{idx3} on column \texttt{b} and \texttt{c} of table \texttt{t} with ascending order.

\begin{code}
CREATE UNIQUE INDEX idx3 ON t(b, c);
\end{code}

Create a UNIQUE BTREE index \texttt{idx4} on column \texttt{a} of table \texttt{t} with ascending order.

\begin{code}
CREATE UNIQUE INDEX idx4 ON t(a);
\end{code}

\textbf{Column store index example}

Create table \texttt{s}, and then add the index \texttt{idx10} on column \texttt{b} of table \texttt{s}.

\begin{code}
CREATE TABLE s (a INT, b NVARCHAR(10), c NVARCHAR(20)); CREATE INDEX idx10 ON s(b);
\end{code}

Create an INVERTED VALUE index \texttt{idx11} on column \texttt{a} of table \texttt{s}.

\begin{code}
CREATE INVERTED VALUE INDEX idx11 ON s(a, b);
\end{code}

Create a unique INVERTED INDIVIDUAL index \texttt{idx12} on column \texttt{a} and \texttt{c} of table \texttt{s}.

\begin{code}
CREATE UNIQUE INVERTED INDIVIDUAL INDEX idx12 ON s(a, c);
\end{code}

Create a UNIQUE INVERTED VALUE index \texttt{idx13} on column \texttt{b} and \texttt{c} of table \texttt{s} and set the load unit type to PAGE LOADABLE.

\begin{code}
CREATE UNIQUE INVERTED VALUE INDEX idx13 ON s(b, c) PAGE LOADABLE;
\end{code}

The base columns \texttt{b} and \texttt{c} allow NULL values. The following inserts are possible:

\begin{code}
INSERT INTO s VALUES(0, 1, NULL);
INSERT INTO s VALUES(1, NULL, NULL);
INSERT INTO s VALUES(2, NULL, NULL);
\end{code}

But the following insert will generate a unique constraint violation error:

\begin{code}
INSERT INTO s VALUES(3, 1, NULL);
\end{code}

\section*{Related Information}

\begin{itemize}
\item ALTER INDEX Statement (Data Definition) [page 452]
\item INDEXES System View [page 1585]
\item INDEX_COLUMNS System View [page 1587]
\item SAP HANA Configuration Parameter Reference
\item M_SYSTEM_LIMITS System View [page 2192]
\end{itemize}
4.10.1.57 CREATE JWT PROVIDER Statement (Access Control)

Defines a JWT provider in the SAP HANA database.

Syntax

```
CREATE JWT PROVIDER <jwt_provider_name>      
  <issuer_clause> <claims_clause>      
  [ <case_clause> ]      
  [ <priority_clause> ]     
  [ <enable_user_creation_clause> ]
```

Syntax Elements

**jwt_provider_name**

Specifies the identifier of a JWT provider to be created. The JWT provider name must not already be defined.

```
<jwt_provider_name> ::= <simple_identifier>
```

**issuer_clause**

Sets JWT provider information.

```
<issuer_clause> ::= WITH ISSUER <issuer_claim>
```

**issuer_claim**

is the issuer name provided in the iss claim of the JWT tokens issued by this JWT provider. <issuer_claim> must be unique across all JWT providers.

```
<issuer_claim> ::= <string_literal>
```

**claims_clause**

```
<claims_clause> ::=      
  <external_id_clause>     
  [ <application_user_clause> ]     
  [ <compare_claim_clause> ]
```

**external_id_clause**

Specifies the claim provided in the JWT tokens to use for mapping the HANA user to an external user name.

```
<external_id_clause> ::= CLAIM <string_literal> AS EXTERNAL IDENTITY
```

**application_user_clause**
The value of this claim in the JWT token is used to set the XS_APPLICATIONUSER session variable during login.

```sql
<application_user_clause> ::= CLAIM <string_literal> AS APPLICATION USER
```

**compare_claim_clause**

A claim can only be used for one compare operation, be it equal or has_member.

```sql
<compare_claim_clause> ::= <compare_claim>[, <compare_claim>[,...]]
```

```sql
<compare_claim> ::= { <claim_equals_clause> | <claim_has_member_clause> }
```

**claim_equals_clause**

```sql
<claim_equals_clause> ::= CLAIM <string_literal> = <string_literal>
```

**claim_has_member_clause**

```sql
<claim_has_member_clause> ::= CLAIM <string_literal> HAS MEMBER <string_literal>
```

**case_clause**

Specifies whether the user mapping is checked case sensitive (default) or insensitive.

```sql
<case_clause> ::= CASE { SENSITIVE | INSENSITIVE } IDENTITY
```

**priority_clause**

Specifies the priority order in which the issuer is checked.

```sql
<priority_clause> ::= PRIORITY <number>
```

<number> is a value between 1-255. The default is 100.

Providers with the same issuer must have different priorities. During authentication, the provider with the highest priority is checked first. If the claims match with the provided JWT token, that provider is used for the authentication; otherwise, the provider with the next lower priority is tested until a match is found.

**enable_user_creation_clause**

Enables automatic user creation when using JWT authentication, and specifies type of user to create - standard or restricted.

```sql
<enable_user_creation_clause> ::= ENABLE USER CREATION [ USER TYPE { STANDARD | RESTRICTED } ] [ USERGROUP <usergroup_name> ] LDAP AUTHORIZATION
```
When enabling automatic user creation, optionally specify the type of the new users to be created. By default, new users are created as STANDARD users. The user is created in the specified user group. The RESTRICTED user created using this clause is not automatically granted the PUBLIC role.

Permissions

Only database users that have the USER ADMIN system privilege can create a JWT provider.

To use the `<enable_user_creation_clause>` also requires the OPERATOR object privilege on the referenced USERGROUP.

Examples

Create a JWT provider with the name my_jwt_provider, using issuer name www/url/my_url and claim user1. The identity checked by the provider is case sensitive.

```sql
CREATE JWT PROVIDER my_jwt_provider WITH ISSUER 'www/url/my_url'
    CLAIM 'user1' AS EXTERNAL IDENTITY CASE SENSITIVE IDENTITY;
```

Create a JWT provider named prov_a with an issuer named xsuaa and three compare claims: origin, aud, and sub.

```sql
CREATE JWT PROVIDER prov_a WITH ISSUER 'http://xsuaa' CLAIM 'origin' = 'http://customerA'
    CLAIM 'aud' HAS MEMBER 'app1'
    CLAIM 'sub' AS EXTERNAL IDENTITY;
```

Create a JWT provider named prov_b with an issuer named xsuaa and two claims: sub and appuser.

```sql
CREATE JWT PROVIDER prov_b WITH ISSUER 'http://xsuaa'
    CLAIM 'sub' AS EXTERNAL IDENTITY
    CLAIM 'appuser' AS APPLICATION USER PRIORITY 110;
```

Related Information

- ALTER JWT PROVIDER Statement (Access Control) [page 455]
- DROP JWT PROVIDER Statement (Access Control) [page 959]
- JWT_PROVIDERS System View [page 1589]
- JWT_USER_MAPPINGS System View [page 1591]
4.10.1.58 CREATE LIBRARY Statement (SQLScript)

Creates a SQLScript user-defined library.

Syntax

```
CREATE [ OR REPLACE ] LIBRARY [ <schema_name>.]<library_name>  
[ LANGUAGE { SQLSCRIPT | SQLSCRIPT TEST } ]  
AS BEGIN    <sqlscript_user_defined_library_spec>    END;
```

Syntax Elements

**OR REPLACE**
Specifying OR REPLACE replaces the existing library, if one exists, with the new definition.

**library_name**
Specifies a name for the new SQLScript user-defined library.

**LANGUAGE**
 Specifies the type of library:

- **SQLSCRIPT**
  Specifies that the library will be a SQLScript library.

- **SQLSCRIPT TEST**
  Specifies that the library will be a SQLScript end-user test framework.

**sqlscript_user_defined_library_spec**
Specifies the body of the new SQLScript user-defined library.

See the SAP HANA SQLScript Reference for the syntax allowed in a SQLScript library.

Description

Use this statement to define a SQLScript library in an SAP HANA database.

A SQLScript user-defined library is a library written in the SQLScript language that can be called by other procedures or functions. A SQLScript library can only be used by SQLScript procedures or functions; it is not available for use in other SQL statements.
Examples

The following example statements create a table, a SQLScript user-defined library that operates on the table, and a procedure that calls the new library.

```sql
CREATE TABLE data_table(col1 INT);
DO BEGIN
  DECLARE idx INT = 0;
  FOR idx IN 1..200 DO
    INSERT INTO data_table VALUES (:idx);
  END FOR;
END;
CREATE OR REPLACE LIBRARY mylib AS BEGIN
  PUBLIC VARIABLE maxval CONSTANT INT = 100;
  PUBLIC FUNCTION bound_with_maxval(i INT) RETURNS x INT AS BEGIN
    x = CASE WHEN :i > :maxval THEN :maxval ELSE :i END;
  END;
  PUBLIC PROCEDURE get_data(IN size INT, OUT result table(col1 INT)) AS BEGIN
    RESULT = SELECT TOP :size col1 FROM data_table;
  END;
END;
CREATE PROCEDURE myproc (IN inval INT) AS BEGIN
  USING mylib AS mylib;
  DECLARE var1 INT = mylib:bound_with_maxval(:inval);
  IF :var1 > mylib:maxval THEN
    SELECT 'unexpected' FROM dummy;
  ELSE
    DECLARE tv table (col1 INT);
    CALL mylib:get_data(:var1, tv);
    SELECT COUNT(*) FROM :tv;
  END IF;
END;
CALL myproc(10);  -- returns the value 10
CALL myproc(150);  -- returns the value 100
```

Related Information

User-Defined Libraries
SAP HANA SQLScript Reference
DROP LIBRARY Statement (SQLScript) [page 961]
ALTER LIBRARY Statement (SQLScript) [page 462]
LIBRARIES System View [page 1600]
LIBRARY_MEMBERS System View [page 1601]
End-User Test Framework in SQLScript
4.10.1.59 CREATE LDAP PROVIDER Statement (Access Control)

Creates an LDAP provider for use with LDAP authorization and authentication.

Syntax

```
CREATE LDAP PROVIDER <ldap_provider_name>   
CREDENTIAL TYPE <credential_type_name> USING <credential_of_ldap_account>   
[ NESTED GROUP LOOKUP URL <url_string_literal> ]   
ATTRIBUTE DN <dn_string_literal>   
[ ATTRIBUTE MEMBER_OF <member_of_string_literal> ]   
[ SSL { ON | OFF } ]   
[ DEFAULT { ON | OFF } ]   
[ { ENABLE | DISABLE } PROVIDER ]   
[ ENABLE USER CREATION FOR LDAP [ <usertype_option> ] ]
```

Syntax Elements

- **ldap_provider_name**
  Specifies the name of an LDAP provider you want to create.

  ```
  <ldap_provider_name> ::= <unicode_name>
  ```

- **CREDENTIAL TYPE credential_type_name USING credential_of_ldap_account**
  Specifies the credential for the LDAP access account. Only one credential type can be configured for an LDAP provider and the only supported type is PASSWORD.

  ```
  <credential_type_name> ::= 'PASSWORD'  
  <credential_of_ldap_account> ::= 'user=<user_dn_string_literal>;password=<passphrase>'
  ```

  LDAP access account information includes the distinguished name (DN) and password of the user that is set up in LDAP server for use by SAP HANA server. This user must have permissions within the LDAP server to perform searches as specified by the USER LOOKUP URL and NESTED GROUP LOOKUP URL clauses.

  For the credential type PASSWORD, specifies the credentials of an LDAP access account by using the USING clause.

  ```
  <credential_of_ldap_account> specifies the credential of the LDAP access account that SAP HANA uses to log in to the LDAP server.
  <user_dn_string_literal> specifies the distinguished name (DN) of the LDAP access account that HANA uses to log in to the LDAP server.
  ```
<passphrase> specifies the password for the access account that SAP HANA uses to log in to the LDAP server.

An example CREDENTIAL clause might look like this: CREDENTIAL TYPE 'PASSWORD' USING 'user=cn=LookupAccount,o=largebank.com;password=secret'.

**USER LOOKUP URL url_string_literal**

Specifies an LDAP URL that locates a unique user entry in the LDAP server that corresponds to an SAP HANA user. The format of a USER LOOKUP URL is as follows:

```
```

A search filter `<filter>` uses the USER_NAME of the SAP HANA user to locate the user entry in the LDAP Server. `<filter>` must include a condition of the form '<attr>=*' where `<attr>` is an LDAP attribute whose value is matched against the USER_NAME of the SAP HANA user. SAP HANA replaces the '*' in the search filter with the USER_NAME of the current SAP HANA user.

For example, when the following LDAP query is sent to the LDAP server, `(&(objectClass=user) (sAMAccountName=*))` is replaced with `(&(objectClass=user) (sAMAccountName=<USER_NAME of the current or specified SAP HANA user>)`:

```
USER LOOKUP URL 'ldap://myhostname:389/ou=Users,dc=largebank,dc=com??sub?(& (objectClass=user) (sAMAccountName=*))'
```

`<filter>` must contain one and only one '*'; otherwise, an error is returned.

The `<attributes>` specification lists LDAP attributes whose values are returned to the SAP HANA server for a given user entry. `<attributes>` must be left empty for USER LOOKUP URL. SAP HANA constructs the `<attributes>` internally based on the LDAP attributes specified with ATTRIBUTE clause.

**NESTED GROUP LOOKUP URL url_string_literal**

Specifies an LDAP URL for obtaining LDAP group membership information, including nested groups, for a user from the LDAP server. For example: NESTED GROUP LOOKUP URL 'ldap://myhostname:389/ou=groupsOU,dc=x??sub?(member:1.2.840.113556.1.4.1941:=*)'

The asterisk in `<url_string_literal>` is replaced by the user DN, which is obtained by using the USER LOOKUP URL.

Depending upon the NESTED GROUP LOOKUP URL, one or more levels of LDAP groups may be returned. It is possible to specify a URL that returns only one level of groups with NESTED GROUP LOOKUP URL. SAP HANA sends this LDAP query to the LDAP server for execution. Before sending the query, SAP HANA replaces '*' in the NESTED GROUP LOOKUP URL with the DN of the user obtained from execution of USER LOOKUP URL.

While ATTRIBUTE MEMBER_OF and NESTED GROUP LOOKUP URL are both optional, if LDAP authorization is used and neither of these are defined, then an error is returned at run-time.

Support for nested LDAP groups is provided only when a single URL can be specified such that it returns a complete list of groups with the user’s membership (including groups with indirect membership). SAP HANA does not recursively fetch nested groups if they were not obtained by executing the search specified by NESTED GROUP LOOKUP URL.

**ATTRIBUTE DN dn_string_literal**

Specifies the LDAP attribute that provides the DN (distinguished name) of the LDAP User entry. The DN of the user is stored in the SAP HANA catalog.
ATTRIBUTE MEMBER_OF member_of_string_literal

Specifies the LDAP attribute that provides a list of groups that a user is a member of. If NESTED GROUP LOOKUP URL is specified, then the group information is obtained using NESTED GROUP LOOKUP URL (that is, ATTRIBUTE MEMBER_OF is not used).

While ATTRIBUTE MEMBER_OF and NESTED GROUP LOOKUP URL are both optional, if LDAP authorization is used and neither of these are defined, then an error is returned at run-time.

SSL {ON | OFF}

Specifies whether SSL/TLS secures connections to the LDAP server, both for LDAP access account authentication and LDAP user and group searches. The default is OFF.

⚠️ Caution

If using the LDAP server for user authorization and authentication, you must secure communication between SAP HANA and the LDAP server using the TLS/SSL protocol to protect the transmission of sensitive information such as user names, passwords, and group membership, which are otherwise sent in the clear between SAP HANA and the LDAP server.

When using SSL protocol, the trust store used to authenticate communication must be a certificate collection in the SAP HANA database with the purpose LDAP. The certificate of the Certificate Authority (CA) that signed the certificate used by the LDAP server must be available in this certificate collection. For more information, see the section on certificate management in the SAP HANA Security Guide.

When set to ON, the SSL/TLS protocol is used and the URL begins with "ldap://".

Connections to LDAP Server can also be secured by using Secure LDAP protocol. In this case, URL begins with "ldaps://". SSL should be set to OFF when using Secure LDAP protocol.

DEFAULT {ON | OFF}

Designates the LDAP provider to use for LDAP authorization and authentication. The default is OFF.

You can create multiple named LDAP providers, but you can only designate one as the default. Designating an LDAP provider as DEFAULT removes the default designation of the previous default LDAP provider, if any.

{ENABLE | DISABLE } PROVIDER

Creates the provider as enabled or disabled. The default is DISABLE PROVIDER.

If a default LDAP provider is disabled, then users cannot log in by using LDAP authorization. VALIDATE LDAP PROVIDER can still be used to verify the configuration of a disabled LDAP provider.

ENABLE USER CREATION FOR

Enables automatic user creation when using LDAP authentication, and specifies type of the user to create - standard or restricted.

```
[ ENABLE USER CREATION FOR LDAP [ <usertype_option> ] ]
<usertype_option> ::= USER TYPE { STANDARD | RESTRICTED }
```

When enabling automatic user creation, optionally specify the type of the new users to be created. By default, new users are created as STANDARD users.
**Description**

To use LDAP group-based authorization or LDAP authentication, access to an LDAP Server is configured in SAP HANA using an LDAP provider.

LDAP authentication and LDAP authorization require a default LDAP provider that is in an enabled state.

Nested LDAP groups is only supported when there is a single URL that can be used to return the complete list of groups that the user is a member of (including groups with indirect membership). Nested LDAP groups is not supported on directory servers that do not provide this capability.

**Permissions**

Only users with the LDAP ADMIN privilege can create LDAP providers.

**Examples**

**Example 1**: Create an LDAP provider, `my_ldap_provider`, for obtaining LDAP group memberships for SAP HANA users and activating it as the default LDAP provider for authorization.

```sql
CREATE LDAP PROVIDER my_ldap_provider
  CREDENTIAL TYPE 'PASSWORD' USING
  'user=cn=LookupAccount,o=largebank.com;password=secret'
  USER LOOKUP URL 'ldap://myhostname:389/ou=Users,dc=largebank,dc=com??sub?'
  (&(objectClass=user)(sAMAccountName=*))
  ATTRIBUTE DN 'distinguishedName'
  ATTRIBUTE MEMBER_OF 'memberOf'
  SSL ON
  DEFAULT ON
  ENABLE PROVIDER;
```

**Example 2**: Create an LDAP provider to use nested groups:

```sql
CREATE LDAP PROVIDER my_ldap_provider
  CREDENTIAL TYPE 'PASSWORD' USING
  'user=cn=LookupAccount,o=largebank.com;password=secret'
  USER LOOKUP URL 'ldap://myhostname:389/ou=Users,dc=largebank,dc=com??sub?'
  (&(objectClass=user)(sAMAccountName=*))
  USER LOOKUP URL 'ldap://myhostname:389/ou=groupsOU,dc=x??sub?'
  (member:1.2.840.113556.1.4.1941:=-*)
  ATTRIBUTE DN 'distinguishedName'
  SSL ON
  DEFAULT ON
  ENABLE PROVIDER;
```

**Example 3**: Create an LDAP provider configured for automatic creation of STANDARD users with LDAP authentication:

```sql
CREATE LDAP PROVIDER my_ldap_provider
  CREDENTIAL TYPE 'PASSWORD' USING
  'user=cn=LookupAccount,o=largebank.com;password=secret'
  USER LOOKUP URL 'ldap://myhostname:389/ou=Users,dc=largebank,dc=com??sub?'
  (&(objectClass=user)(sAMAccountName=*))
```

SAP HANA SQL Reference Guide for SAP HANA Platform
Example 4: Configure an LDAP Provider for automatic creation of RESTRICTED users with LDAP authentication:

```sql
CREATE LDAP PROVIDER my_ldap_provider
  CREDENTIAL TYPE 'PASSWORD' USING
    'user=cn=LookupAccount,o=largebank.com;password=secret'
  USER LOOKUP URL 'ldap://myhostname:389/ou=Users,dc=largebank,dc=com??sub?
    (&(objectClass=user)(sAMAccountName=*))'
  ATTRIBUTE DN 'distinguishedName'
  SSL ON
  DEFAULT ON
  ENABLE PROVIDER
  ENABLE USER CREATION FOR LDAP
  USER TYPE RESTRICTED;
```

Related Information

Certificate Management in SAP HANA
ALTER LDAP PROVIDER Statement (Access Control) [page 458]
DROP LDAP PROVIDER Statement (Access Control) [page 962]
VALIDATE LDAP PROVIDER Statement (Access Control) [page 1195]
LDAP_PROVIDER_URLS System View [page 1599]
LDAP_PROVIDERS System View [page 1597]
LDAP_USERS System View [page 1599]

4.10.1.60 CREATE PROCEDURE Statement (Procedural)

Creates a procedure that uses the specified programming language.

Syntax

```sql
CREATE [ OR REPLACE ] PROCEDURE <proc_name>
  [ (<input_output_parameter_clause>) ]
  [ LANGUAGE <lang> ]
  [ SQL SECURITY [ DEFINER | INVOKER ] ]
  [ DEFAULT SCHEMA <default_schema_name> ]
  [ READS SQL DATA [ WITH RESULT VIEW <view_name> ] ]
  [ <variable_cache_clause> ]
  [ DETERMINISTIC ]
  [ WITH ENCRYPTION ]
  [ AUTOCOMMIT DDL { ON | OFF } ]
  AS
  { BEGIN [ SEQUENTIAL EXECUTION ]
```
Syntax Elements

**proc_name**

The procedure name, with optional schema name.

```plaintext
<proc_name> ::= [ <schema_name>.]<identifier>  
<schema_name> ::= <unicode_name>  
```

**input_output_parameter_clause**

Specifies the input and output parameters for the procedure.

```plaintext
<input_output_parameter_clause> ::= <parameter> [,...]  
<parameter> ::= [ IN | OUT | INOUT ] <param_name> ... [ DEFAULT <constant> ] [ ARRAY ]  
<data_or_table_type> ::= <sql_type> [ DEFAULT <constant> ] [ ARRAY ]  
<table_type>  
<table_type_definition>  
<any_table_type>  
```

The default `<parameter>` is IN. Input and output parameters must be explicitly typed; un-typed tables are not supported.

The input and output parameters of a procedure can have any of the primitive SQL types or a table type. INOUT parameters can only be of scalar type.

**sql_type [ DEFAULT constant ] [ ARRAY ]**

Specifies the data type of the variable.

```plaintext
<sql_type> ::=  
DATE  
TIME  
SECONDDATE  
TIMESTAMP  
TINYINT  
SMALLINT  
INTEGER  
BIGINT  
SMALLDECIMAL  
REAL  
DOUBLE  
BINTEXT  
VARCHAR [ (unsigned_integer) ]  
NVARCHAR [ (unsigned_integer) ]  
ALPHANUM [ (unsigned_integer) ]  
VARBINARY [ (unsigned_integer) ]  
DECIMAL [ (unsigned_integer) [, unsigned_integer] ]  
```

Only for use with IN parameters; use the optional `DEFAULT` option when you want to specify a default and a constant is correct for the data type. For example, you cannot specify a constant as a default for an ARRAY.

Use the optional ARRAY option when the parameter is an array.
**table_type**

Specifies a table type that was previously defined with the CREATE TYPE statement.

```
<table_type> ::= [ <schema_name>.]<identifier>  
<schema_name> ::= <unicode_name>
```

**table_type_definition**

Specifies a table type that is implicitly defined within the signature.

```
<table_type_definition> ::= TABLE (<column_list_definition>)  
<column_list_definition> ::= <column_elem>[,...]  
<column_elem> ::= <column_name> <data_type>  
<column_name> ::= <identifier>
```

**any_table_type**

Defines the table type at DDL time or query compilation time.

```
<any_table_type> ::= TABLE(...)  
```

The syntax of `<any_table_type>` requires you to explicitly specify TABLE(…) (for example, CREATE PROCEDURE anyproc_in (IN itab TABLE(...), outtab TABLE(...)) AS BEGIN...).

**LANGUAGE**

Specifies the programming language that is used in the procedure.

```
LANGUAGE <lang>  
<lang> ::= { SQLSCRIPT | R | GRAPH }
```

The default is SQLSCRIPT. Define the language in all procedure definitions.

Specify GRAPH to use graph script, a domain-specific programming language for custom graph algorithms and analytics.

**SQL SECURITY { DEFINER | INVOKER }**

Specifies the security mode for the procedure. The default is DEFINER.

- **DEFINER**
  Specifies that the procedure is executed with the privileges of the user who defined the procedure.
- **INVOKER**
  Specifies that the procedure is executed with the privileges of the user who invoked the procedure.

**DEFAULT SCHEMA**

Specifies the schema for unqualified objects in the procedure body.

```
DEFAULT SCHEMA <default_schema_name>  
<default_schema_name> ::= <identifier>
```

If nothing is specified, then the current_schema of the session is used.

**READS SQL DATA**

Specifies that the procedure is read only and side-effect free. That is, the procedure does not make modifications to the database data or its structure, even if the procedure body contains dynamic SQL calls.

The advantage of using this parameter is that certain optimizations are available for read-only procedures.
WITH RESULT VIEW

Specifies the result view to use as the output of a read-only procedure.

WITH RESULT VIEW <view_name>

<view_name> ::= <identifier>

When a result view is defined for a procedure, it can be called by an SQL statement in the same way as a table or view. See Example 2 - Using a result view [page 792].

SEQUENTIAL EXECUTION

Forces sequential execution of the procedure logic. No parallelism takes place.

variable_cache_clause

Caches variable values during execution time so that they do not need to be re-resolved as the operations in <statement_body> are executed. Caching a variable removes the risk that its value will change each time it is resolved.

<variable_cache_clause> ::= VARIABLE CACHE ON {
<variable_entry_with_mode_list> | <variable_entry_without_mode_list> }
<variable_entry_with_mode_list> ::= <variable_entry_with_mode> [, ...
<variable_entry_with_mode> ::= <variable_entry enable_mode>
<variable_entry> ::= { <variable_name> | <variable_name_clustered_list> }
<variable_name_clustered_list> ::= ( <variable_name> [, <variable_name> ...
<enable_mode> ::= { ENABLE | DISABLE }
<variable_entry_without_mode_list> ::= <variable_entry> [, <variable_entry> ...

DETERMINISTIC

DETERMINISTIC is for use with SQLScript procedures that produce scalar results. It specifies that the procedure returns the same result any time that it is called with the same input parameters.

WITH ENCRYPTION

Encrypts the procedure definition. See the SAP HANA SQLScript Guide for more information on this clause and the impact of encrypting a procedure.

AUTOCOMMIT DDL { ON | OFF }

Specifies whether to automatically commit DDL statements in the procedure body. Specify ON when automatic commit of DDL statements is required; for example, during administrative operations such as importing data.

This clause is only supported for SQLScript procedures that are not read-only. The default value is OFF.

statement_body

<statement_body> ::= [
<proc_using_list> ] [ <proc_decl_list> ] [ <proc_handler_list> ]
<proc_stmt_list>

Defines the main body of the procedure according to the specified programming language.

proc_using_list

Defines the libraries to use.

<proc_using_list> ::= <proc_using> [ ... ]
<proc_using> ::= USING <library_name> AS <library_alias>;  
<library_name> ::= [ <schema_name>.]<identifier>  
<library_alias> ::= <identifier>

proc_decl_list
Defines the declarations for the procedure.

<proc_decl_list> ::= <proc_declaration> [ <proc_declaration> [ ... ] ]
<proc_declaration> ::=  DECLARE {   <proc_variable>    | <proc_table_variable>   | <proc_row_variable>   | <proc_cursor>   | <proc_condition> };

proc_variable
Defines variables for the procedure.

<proc_variable>::=   <variable_name_list> [ CONSTANT ] { <sql_type> | <array_datatype> | AUTO } [ NOT NULL ] [ <default_value> ]   <variable_name_list> ::= <variable_name>, <variable_name> [,... ]  { DEFAULT | := | } { <value> | <expression> }   <value> ::= A value of the type specified by <sql_type> or an expression
When you specify AUTO, you must also specify <default_value>.

proc_table_variable
Defines table variables for the procedure.

<proc_table_variable> ::= <proc_table_var_def> [ <sort_def> ]
<proc_table_var_def>
<variable_name_list> <table_type_definition> 
| <table_type> 
| <variable_name_list> [ CONSTANT ] LIKE { <table_name> 
| <table_variable_name>.<column_name> | AUTO } [ NOT NULL ] [ <default_value> ]
| [ <variable_name_list> [ CONSTANT ] TABLE LIKE { <table_name> 
| :<table_variable_name> } | AUTO ] [ <default_value> ]
| <sort_def> ::= SEARCH KEY ( <col_name> [, <col_name> [,... ] ] )
When you specify AUTO, you must also specify <default_value>.

<sort_def> is used in SQLScript procedures to sort the table variables. See the SAP HANA SQLScript Guide for more information on sorted table variables.

proc_row_variable
Defines row variables for the procedure.

<proc_row_variable> ::= 
| <variable_name_list> ROW <row_definition> 
| <variable_name_list> ROW LIKE { <persistent_table_name> 
| :<other_table_row_or_cursor_variable_name>
proc_cursor
Defines the procedure cursors.

```sql
<proc_cursor> ::= CURSOR <cursor_name> [ ( <proc_cursor_param_list> ) ] FOR <subquery>;
<proc_cursor_param_list> ::= <proc_cursor_param> [,...]
<variable_name> ::= <identifier>
<cursor_name> ::= <identifier>
<proc_cursor_param> ::= <param_name> <datatype>
```

**proc_condition**

Defines variable conditions.

```sql
<proc_condition> ::= <variable_name> CONDITION
| <variable_name> CONDITION FOR <sql_error_code>
```

**proc_handler_list**

Declares exception handlers to catch SQL exceptions.

```sql
<proc_handler_list> ::= <proc_handler> [,<proc_handler> [...]]
<proc_handler> ::= DECLARE { EXIT | CONTINUE } HANDLER FOR <proc_condition_value_list> <proc_stmt>;
```

**proc_condition_value_list**

Specifies one or more condition values.

```sql
<proc_condition_value_list> ::= <proc_condition_value> [,<proc_condition_value> [...]]
```

**proc_condition_value**

Specifies a specific error code number or condition name declared on a condition variable.

```sql
<proc_condition_value> ::= SQLEXCEPTION | SQLWARNING | <sql_error_code> | <condition_name>
```

**proc_stmt_list**

Specifies statements for the procedure body.

```sql
<proc_stmt_list> ::= <proc_stmt> [,...]
<proc_stmt> ::= <proc_block>
| <proc_assign>
| <proc_tabvar_search>
| <proc_single_assign>
| <proc_multi_assign>
| <proc_if>
| <proc_loop>
| <proc_while>
| <proc_for>
| <proc_foreach>
| <proc_exit>
| <proc_continue>
| <proc_signal>
| <proc_resignal>
| <proc_open>
```
**proc_block**

Nests a section of the procedure by using the BEGIN and END keywords.

```
<proc_block> ::=  
BEGIN <proc_block_option>  
[ <proc_decl_list> ]  
[ <proc_handler_list> ]  
<proc_stmt_list>  
END;
```

The autonomous transaction is independent from the main procedure. Changes made and committed by an autonomous transaction can persist regardless of whether the main procedure transaction is committed or rolled back. The end of the autonomous transaction block has an implicit commit.

**proc_assign**

Assigns values to variables.

```
<proc_assign> ::=  
<variable_name> := { <expression> | <array_function> };  
| <variable_name> '[<expression>]' := <expression>;
```

*<expression>* is either a simple expression such as a character, a date, or a number, or it can be a scalar function.

**proc_tabvar_search**

Searches for key value pairs in a table variable.

The position of the first matching record is returned, or NULL if no record matches. This result can be used in conjunction with other table variable operators (DELETE, UPDATE).

```
<proc_tabvar_search> ::=  
POS = <table_variable>.SEARCH( ( <column_list> ), ( <value_list> ) [<start_position> ] )  
<column_list> ::= <column1> [, <column2> [,..] ]  
<value_list> ::= <value1> [, <value2> [,..] ]  
<start_position> ::= <expression>
```

POS= is part of the required syntax. For example, specify `POS=:myTabVar.SEARCH(('key1', "key2"), ('I', 3));` to search for values "I" and 3 in columns key1 and key2, respectively, of the myTabVar table variable.

**table_variable**

Specifies the table variable to search.

**column_list and value_list**
Specifies the values to search for in the columns. The SAP HANA server pairs the columns listed with the values listed, thus forming the key-value pairs to search for. The number of items in `<column_list>` must be the same as the number of items in `<value_list>`.

A column can be listed more than once in `<column_list>`, just as a value can be listed more than once in `<value_list>`.

`start_position` specifies the row to start with. The default value is 1, which specifies to scan all data starting at the beginning row. `<expression>` can be any valid expression that resolves to a positive integer.

`array_function` specifies an array function.

```plaintext
<array_function> ::= 
  ARRAY_AGG ( <table_variable>.<column_name> [ ORDER BY <sort_spec_list> ] ) 
  | CARDINALITY ( <array_variable_name> ) 
  | TRIM_ARRAY ( <array_variable_name>,<array_variable_name> ) 
  | ARRAY ( <array_variable_name_list> ) 

<table_variable> ::= <identifier> 
<column_name> ::= <identifier> 
<array_variable_name> ::= <identifier>
```

The `ARRAY_AGG` function returns the array by aggregating the set of elements in the specified column of the table variable. Ordering the elements is optional.

The `CARDINALITY` function returns the number for the element in the array `<array_variable_name>`.

The `TRIM_ARRAY` function returns the new array by removing the given number of elements, `<numeric_value_expression>`, from the end of the array, `<array_value_expression>`.

The `ARRAY` function returns an array whose elements are specified in the list `<array_variable_name>`. For more information, see SAP HANA SQLScript Reference.

`proc_single_assign` assigns values to a list of variables with only one function evaluation.

```plaintext
<proc_single_assign> ::= 
  <variable_name> = <subquery> 
  |  <variable_name> = <proc_ce_call> 
  |  <variable_name> = <proc_apply_filter> 
  |  <variable_name> = <unnest_function> 
  |  <variable_name> = <match_merge_op>
```

`proc_multi_assign` assigns values to a list of variables with multiple function evaluations.

```plaintext
<proc_multi_assign> ::= ( <variable_name_list> ) := <function_expression>
```

`<function_expression>` is a scalar user-defined function and the number of elements in `<variable_name_list>` equal the number of output parameters of the scalar user-defined function.

`proc_ce_call` defines the calculation engine plan operators for the procedure.

Use SQL rather than calculation engine plan operators with SQLScript.
## Overview: Mapping between CE_* Operators and SQL

<table>
<thead>
<tr>
<th>CE Operator</th>
<th>CE Syntax</th>
<th>SQL Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE_COLUMN_TABLE</td>
<td>CE_COLUMN_TABLE(&lt;table_name&gt;[,&lt;attributes&gt;])</td>
<td>SELECT [&lt;attributes&gt;] FROM &lt;table_name&gt;</td>
</tr>
<tr>
<td>CE_JOIN_VIEW</td>
<td>CE_JOIN_VIEW(&lt;column_view_name&gt;[,&lt;attributes&gt;])</td>
<td>SELECT [&lt;attributes&gt;] FROM &lt;column_view_name&gt; out = CE_JOIN_VIEW(&quot;PRODUCT_SALES&quot;, [&quot;PRODUCT_KEY&quot;, &quot;PRODUCT_TEXT&quot;, &quot;SALES&quot;]);</td>
</tr>
<tr>
<td>CE_Olap_VIEW</td>
<td>CE_Olap_VIEW(&lt;olap_view_name&gt;[,&lt;attributes&gt;])</td>
<td>SELECT [&lt;attributes&gt;] FROM &lt;olap_view_name&gt; out = CE_Olap_VIEW(&quot;OLAP_view&quot;, [&quot;DIM1&quot;, SUM(&quot;KF&quot;)]);</td>
</tr>
<tr>
<td>CE_CALC_VIEW</td>
<td>CE_CALC_VIEW(&lt;calc_view_name&gt;[,&lt;attributes&gt;])</td>
<td>SELECT [&lt;attributes&gt;] FROM &lt;calc_view_name&gt; out = CE_CALC_VIEW(&quot;TESTCECTABLE&quot;, [&quot;CID&quot;, &quot;CNAME&quot;]);</td>
</tr>
<tr>
<td>CE_JOIN</td>
<td>CE_JOIN(&lt;left_table&gt;,&lt;right_table&gt;[&lt;join_attributes&gt;[&lt;projection_list&gt;]])</td>
<td>SELECT [&lt;projection_list&gt;] FROM &lt;left_table&gt;,&lt;right_table&gt; WHERE &lt;join_attributes&gt; ot_pubs_books1 = CE_JOIN (:lt_pubs, :it_books, [&quot;PUBLISHER&quot;]);</td>
</tr>
<tr>
<td>CE_LEFT_OUTER_JOIN</td>
<td>CE_LEFT_OUTER_JOIN(&lt;left_table&gt;,&lt;right_table&gt;[&lt;join_attributes&gt;[&lt;projection_list&gt;]])</td>
<td>SELECT [&lt;projection_list&gt;] FROM &lt;left_table&gt; LEFT OUTER JOIN &lt;right_table&gt; ON &lt;join_attributes&gt; ot_pubs_books1 = SELECT P.publisher AS publisher, name, street, post_code, city, country, isbn, title, edition, year, price, csrcy FROM :lt_pubs AS P, :it_books AS B WHERE P.publisher = B.publisher;</td>
</tr>
<tr>
<td>CE Operator</td>
<td>CE Syntax</td>
<td>SQL Equivalent</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>CE_RIGHT_OUTER_JOIN</strong></td>
<td>CE_RIGHT_OUTER_JOIN(&lt;left_table&gt;, &lt;right_table&gt;, &lt;join_attributes&gt;[&lt;projection_list&gt;])</td>
<td>SELECT [&lt;projection_list&gt;] FROM &lt;left_table&gt; RIGHT OUTER JOIN &lt;right_table&gt; ON &lt;join_attributes&gt;</td>
</tr>
<tr>
<td><strong>CE_PROJECTION</strong></td>
<td>CE_PROJECTION(&lt;table_variable&gt;, &lt;projection_list&gt; [[, &lt;filter&gt;]])</td>
<td>SELECT &lt;projection_list&gt; FROM &lt;table_variable&gt; where [&lt;filter&gt;]</td>
</tr>
<tr>
<td></td>
<td>ot_books1 = CE_PROJECTION(:it_books, [&quot;TITLE&quot;, &quot;PRICE&quot;, &quot;CRCY&quot; AS &quot;CURRENCY&quot;], 'PRICE' &gt; 50);</td>
<td>ot_book2 = SELECT title, price, crcy AS currency FROM :it_books WHERE price &gt; 50;</td>
</tr>
<tr>
<td><strong>CE_UNION_ALL</strong></td>
<td>CE_UNION_ALL(&lt;table_variable1&gt;,&lt;table_variable2&gt;)</td>
<td>SELECT * FROM &lt;table_variable1&gt; UNION ALL SELECT * FROM &lt;table_variable2&gt;</td>
</tr>
<tr>
<td></td>
<td>ot_all_books1 = CE_UNION_ALL(:lt_books, :it_audiobooks);</td>
<td>ot_all_books2 = SELECT * FROM :lt_books UNION ALL SELECT * FROM :it_audiobooks;</td>
</tr>
<tr>
<td><strong>CE_CONVERSION</strong></td>
<td>CE_CONVERSION(&lt;table_variable&gt;, &lt;conversion_params&gt; [,&lt;rename_clause&gt;])</td>
<td>SQL-Function CONVET_CONVERT_CURRENCY</td>
</tr>
<tr>
<td><strong>CE_AGGREGATION</strong></td>
<td>CE_AGGREGATION(&lt;table_variable&gt;, &lt;aggregate_list&gt; [,&lt;group_columns&gt;])</td>
<td>SELECT &lt;aggregate_list&gt; FROM &lt;table_variable&gt; [GROUP BY &lt;group_columns&gt;]</td>
</tr>
<tr>
<td></td>
<td>ot_books1 = CE_AGGREGATION(:it_books, [COUNT (&quot;PUBLISHER&quot;) AS &quot;CNT&quot;], [&quot;YEAR&quot;]);</td>
<td>ot_books2 = SELECT COUNT (publisher) AS cnt, year FROM :it_books GROUP BY year;</td>
</tr>
<tr>
<td><strong>CE_CALC</strong></td>
<td>CE_CALC('&lt;expr&gt;', &lt;result_type&gt;)</td>
<td>SQL Function TEMPL = SELECT &quot;ID&quot; AS &quot;KEY&quot;, ROW_NUMBER() OVER () AS &quot;T_ID&quot; FROM :table_var</td>
</tr>
<tr>
<td></td>
<td>TEMP = CE_PROJECTION(:table_variable, [&quot;ID&quot; AS &quot;KEY&quot;, CE_CALC('rownum()', INTEGER) AS &quot;T_ID&quot;);</td>
<td>TEMP = CE_PROJECTION(:table_variable, [&quot;ID&quot; AS &quot;KEY&quot;], CE_CALC('rownum()', INTEGER) AS &quot;T_ID&quot;);</td>
</tr>
</tbody>
</table>

SAP HANA SQL Reference Guide for SAP HANA Platform

SQL Reference

PUBLIC

785
<table>
<thead>
<tr>
<th>CE Operator</th>
<th>CE Syntax</th>
<th>SQL Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE_VERTICAL_UNION</td>
<td>CE_VERTICAL_UNION(&lt;table_variable&gt;, &lt;projection_list&gt;[ {,&lt;table_variable&gt;, &lt;projection_list&gt;}...] )</td>
<td>unnest ( &lt;table_variable&gt; [ {, &lt;table_variable&gt; }_] ) as ( (&lt;projection_list&gt;) [ { , ( &lt;projection_list&gt; ) } ... ] )</td>
</tr>
</tbody>
</table>

For more information about calculation engine plan operators, see SAP HANA SQLScript Reference.

**proc_apply_filter**

Defines a dynamic WHERE condition that is applied at runtime.

<proc_apply_filter> ::= APPLY_FILTER ( {<table_name> | <table_variable>}, <variable_name> );

For more information about APPLY_FILTER, see SAP HANA SQLScript Reference.

**unnest_function**

Returns a table, including a row for each element of the specified array.

<unnest_function> ::= UNNEST ( <variable_name_list> ) [ WITH ORDINALITY ] [ <as_col_names> ];

**WITH ORDINALITY**

Appends an ordinal column to the return values.

**as_col_names**

Specifies the column names of the return table.

<as_col_names> ::= AS [ <table_name> ] ( <column_name_list> )
<column_name_list> ::= <column_name> [, <column_name> [,...] ]

**map_merge_op**

Applies each row of the input table to the mapper function and unites all intermediate result tables. For more information about the MAP_MERGE operator, see the SAP HANA SQLScript Reference.

<map_merge_op> ::= MAP_MERGE ( <table_or_table_variable>, <mapper_identifier> ( <table_or_table_variable>.<column_name> [,,...] [ , <param_list> ] )
<table_variable_name> ::= <identifier>
<mapper_identifier> ::= <identifier>
<column_name> ::= <identifier>
<param_list> ::= <param> [,,...]
<param> ::= <table_or_table_variable>
<string_literal> | <numeric_literal>
<identifier>
proc_if

Controls the execution flow with conditionals.

```
<proc_if> ::= 
  IF <condition> THEN [ SEQUENTIAL EXECUTION ] 
  [ [ <proc_decl_list> ] ] 
  [ <proc_stmt_list> ] 
  [ <proc_handler_list> ] 
  [ <proc_elsif_list> ] 
  [ <proc_else> ] 
  END IF;
<proc_elsif_list> ::= 
  ELSEIF <proc_decl_list> ] [ <proc_handler_list> ] <proc_stmt_list> 
<proc_else> ::= 
  ELSE [ SEQUENTIAL EXECUTION ] [ <procDecl_list> ] [ <proc_handler_list> ] 
  <proc_stmt_list>
```

proc_loop

Uses a loop to repeatedly execute a set of statements.

```
<proc_loop> ::= 
  LOOP [ SEQUENTIAL EXECUTION ] 
  [ [ <proc_decl_list> ] ] 
  [ <proc_handler_list> ] 
  <proc_stmt_list> 
END LOOP;
```

proc_while

Specifies that a set of trigger statements is repeatedly called while a condition is true.

```
<proc_while> ::= 
  WHILE <condition> DO [ SEQUENTIAL EXECUTION ] 
  [ [ <proc_decl_list> ] ] 
  [ <proc_handler_list> ] 
  <proc_stmt_list> 
END WHILE;
```

proc_for

Specifies a FOR...IN loop that iterates over a set of data.

```
<proc_for> ::= 
  FOR <column_name> IN [ REVERSE ] <expression> [ ... ] <expression> 
  DO [ SEQUENTIAL EXECUTION ] 
  [ [ <proc_decl_list> ] ] 
  [ <proc_handler_list> ] 
  <proc_stmt_list> 
END FOR;
```

proc_foreach

Specifies a FOR...EACH loops that iterates over all elements in a set of data.

```
<proc_foreach> ::= 
  FOR <column_name> AS <column_name> [ [ open_param_list> ] 
  DO [ SEQUENTIAL EXECUTION ] 
  [ [ <proc_decl_list> ] ] 
  [ <proc_handler_list> ] 
  <proc_stmt_list> 
END FOR;
```
<open_param_list> ::= ( <expression> [, <expression> [,... ] )

**proc_exit**
Terminates a loop.

<proc_exit> ::= BREAK;

**proc_continue**
Skips a current loop iteration and continues with the next value.

<proc_continue> ::= CONTINUE;

**proc_signal**
Explicitly raises an exception from within your trigger procedures.

<proc_signal> ::=   SIGNAL <signal_value> [ <set_signal_info> ];

**proc_resignal**
Raises an exception on the action statement in an exception handler.

<proc_resignal> ::=   RESIGNAL [ <signal_value> ] [ <set_signal_info> ];

If an error code is not specified, then RESIGNAL throws the caught exception.

**signal_value**
Specifies a SIGNAL or RESIGNAL for a signal name or an SQL error code.

<signal_value> ::= <signal_name> | <sql_error_code>
<signal_name> ::= <identifier>
<sql_error_code> ::= <unsigned_integer>

**set_signal_info**
Specifies that an error message is delivered to users when a specified error is thrown during procedure execution.

<set_signal_info> ::= SET MESSAGE_TEXT = '<message_string>'
<message_string> ::= <any_character>

**proc_sql**
Specifies an SQL statement.

<proc_sql> ::=<subquery>
  | <select_into_stmt>
  | <select_from_embedded_function_stmt>
  | <insert_stmt>
  | <delete_stmt>
  | <update_stmt>
  | <replace_stmt>
  | <call_stmt>
  | <create_table_stmt>
  | <drop_table_stmt>
<select_from_embedded_function_stmt> and <select_into_stmt> are for use exclusively in a SQLScript procedure body context. For all other SQL statements in <proc_sql>, refer to the topic for that specific statement found in this guide. For example, for more information on <insert_stmt>, see the INSERT Statement topic in this guide.

**select_from_embedded_function_stmt**

Selects from an embedded function.

```sql
<select_from_embedded_function> ::= SELECT <select_list> FROM SQL FUNCTION <embedded_func_returns> BEGIN <sqlscript_body> END
<embedded_func_returns> = RETURNS { <table_ref> | ( <opt_cv_array_column_list> ) | <proc_param_name> <func_data_type> }
```

Any query with SQL FUNCTION in the body is not cached in SQL plan cache.

Queries with SQL FUNCTION can be used in SELECT or DML statements, not inside a DDL definition, and not inside a DO BEGIN...END or another SQL FUNCTION block.

For examples of how to use embedded SQL functions in the body of a SQLScript procedure, see the SAP HANA SQLScript reference guide.

**select_into_stmt**

Specifies a query.

```sql
<select_into_stmt> ::= SELECT <select_list> INTO <variable_name_list> [ DEFAULT <scalar_expr_list> ] <from_clause> [ <where_clause> ] [ <group_by_clause> ] [ <having_clause> ] [ {<set_operator> <subquery>, ... } ] [ <order_by_clause> ] [ <limit> ];
```

For information on <select_list>, <from_clause>, <where_clause>, <group_by_clause>, <having_clause>, <set_operator>, <subquery>, <order_by_clause>, and <limit>, see the SELECT statement.

**variable_name_list**

Specifies a variable list.

```sql
<variable_name_list> ::= <variable_name>[(, <variable_name>)...]
<variable_name> ::= <identifier>
```

**<variable_name>** is a scalar variable. You can assign a selected item value to this scalar variable.

**proc_open**

Controls cursor operations.

```sql
<proc_open> ::= OPEN <cursor_name> [ <open_param_list> ];
<proc_fetch> ::= FETCH <cursor_name> INTO <column_name_list>;
<proc_close> ::= CLOSE <cursor_name>;
```
Calls a procedure using the CALL statement. See CALL Statement (Procedural) [page 715].

**proc_exec**

Makes a dynamic SQL call.

```plaintext
<proc_exec> ::= EXEC '<sql-statement>' [ INTO <variable_name_list> ] [ DEFAULT <scalar_expr_list> ] [ USING <expression_list> ] [ EXECUTE IMMEDIATE '<sql-statement>' [ INTO <variable_name_list> ] [ DEFAULT <scalar_expr_list> ] [ USING <expression_list> ] [ READS SQL DATA ];
<variable_name_list> ::= <variable_name>[,...]
<variable_name> ::= <identifier>
<expression_list> ::= <expression>[,...]
```

EXEC executes *<sql-statement>* passed as a string argument. EXEC does not return a result set if *<sql-statement>* is a SELECT statement. Use EXECUTE IMMEDIATE for that purpose. If the query returns result sets or output parameters, then you can assign the values to scalar or table variables with the INTO clause.

When *<sql-statement>* is a SELECT statement and table variables are listed in the INTO clause, result sets are assigned to the table variables sequentially. If scalar variables are listed in the INTO clause for a SELECT statement, then it works like *<select_into_stmt>* and assigns the value of each column of the first row to a scalar variable when a single row is returned from a single result set. When the SQL statement is a CALL statement, output parameters represented as ':<variable_name>' in the SQL statement are assigned to the variables in the INTO clause according to the variable name.

If *<sql-statement>* returns a single row, then you can assign the value of each column to a scalar variable by specifying the INTO *<variable_name_list>* clause.

Bind scalar values by specifying the USING *<expression_list>* clause. *<expression>* can be either a simple expression, such as a character, a date, or a number, or it can be a scalar variable.

READS SQL DATA specifies that the dynamic SQL call is read only and will not make modifications to the database data or its structure.

**proc_return**

Returns a value from a procedure.

```plaintext
<proc_return> ::= RETURN [ <proc_expr> ];
```

For more information about SQLScript, see SAP HANA SQLScript Reference.

For more information about the R-Language, see the SAP HANA R Integration Guide.

**proc_insert**

Inserts a new data record into a table variable at a specific position.

```plaintext
<proc_insert> ::= :<table_variable>.INSERT((<value>[,...] ), <index>)
```

All existing data records at the positions starting from the given index are moved to the next position. If the index is greater than the original table size, then the records between the inserted record and the original last record are initialized with NULL values.

**proc_update**
Updates a data record at a specific position in a table variable. There are two syntaxes for `<proc_update>`:

```
<proc_update> ::= :<table_variable>.UPDATE({<value>[,]...}, <index>)
<proc_update> ::= :<table_variable>[<index>] = {<value>[,]...}
```

If `<index>` is greater than the table size, then no operation is performed.

**proc_delete**

Deletes a data record at a specific position in a table variable.

```
<proc_delete> ::= :<table_variable>.DELETE(<index>)
```

If `<index>` is greater than the table size, then no operation is performed.

**HEADER ONLY**

Specifies that only the properties of the procedure are created along with the OID and no procedure dependencies exist.

Once the headers are replaced with full procedure definitions, dependencies are created while the procedure OID remains the same. Dependencies appear in the OBJECT_DEPENDENCIES system view. **HEADER ONLY** is useful when several dependent procedures must be created.

**Description**

The `CREATE PROCEDURE` statement creates a procedure using the specified programming language `<lang>`.

For more information about SQLScript, see *SAP HANA SQLScript Reference*.

For more information about the R-Language, see the *SAP HANA R Integration Guide*.

**CREATE OR REPLACE behavior:**

- The behavior of CREATE OR REPLACE depends on the existence of a pre-defined procedure: If there is no existing procedure, then it behaves like `CREATE PROCEDURE`; otherwise, it behaves like `ALTER PROCEDURE`.
- Explicitly set all parameters where the desired setting is different from the default (for example, settings for the procedure language, the default schema, whether the procedure is deterministic or encrypted, sequential execution options, and so on). Everything not specified in a `CREATE OR REPLACE` statement is reset to its default.
- You cannot use `CREATE OR REPLACE` to change the route options or force a recompile of an existing procedure.
- If you specify the security mode or route options, then the behavior depends on the existence of a pre-defined procedure: If there is no existing procedure, then the operation succeeds (the procedure is created). If the procedure exists, and the security mode or route option is different, then an error is returned.
- `CREATE OR REPLACE` does not change any DDL settings for the procedure, such as grant privileges, dependent objects, and so on.
Examples

Example - Creating an SQL Script Procedure

Create an SQLScript procedure with the following definition.

```
CREATE PROCEDURE orchestrationProc  LANGUAGE SQLSCRIPT AS
BEGIN
  DECLARE v_id BIGINT;
  DECLARE v_name VARCHAR(30);
  DECLARE v_pmnt BIGINT;
  DECLARE v_msg VARCHAR(200);
  DECLARE CURSOR c_cursor1 (p_payment BIGINT) FOR
    SELECT id, name, payment FROM control_tab
    WHERE payment > :p_payment ORDER BY id ASC;
  CALL init_proc();
  OPEN c_cursor1(250000);
  FETCH c_cursor1 INTO v_id, v_name, v_pmnt;
  v_msg := :v_name || ' (id ' || :v_id || ') earns ' || :v_pmnt || ' $.';
  CALL ins_msg_proc(:v_msg);
  CLOSE c_cursor1;
END;
```

The procedure features a number of imperative constructs, including the use of a cursor (with associated state) and local scalar variables with assignments.

Example - Using a result view

The following example creates an SQLScript procedure that uses a result view ProcView to return its results.

```
CREATE PROCEDURE ProcWithResultView(IN id INT, OUT o1 CUSTOMER)  LANGUAGE SQLSCRIPT
  READS SQL DATA WITH RESULT VIEW ProcView AS
BEGIN
  o1 = SELECT * FROM CUSTOMER WHERE CUST_ID = :id;
END;
```

You call this procedure from an SQL statement as follows.

```
SELECT * FROM ProcView (PLACEHOLDER."id"=>'5');
```

Procedures and result views produced by procedures are not connected from the security perspective and therefore do not inherit privileges from each other. The security aspects of each object must be handled separately. For example, you must grant the SELECT privilege on a result view and EXECUTE privilege on a connected procedure.

Create the procedure ProcHeader as HEADER only. Later, you use ALTER PROCEDURE to replace the header with the full procedure definitions.

```
CREATE PROCEDURE ProcHeader (IN id INTEGER) AS HEADER ONLY;
ALTER PROCEDURE ProcHeader (IN id INTEGER) AS
BEGIN
  o1 = SELECT * FROM CUSTOMER WHERE CUST_ID = :id;
END;
```

Example - Using variable cache
The following example creates an SQLScript procedure that caches variable X:

```
CREATE TABLE test_cache (a INT);
INSERT INTO test_cache VALUES(5);
CREATE OR REPLACE PROCEDURE test_result_cache
  LANGUAGE SQLSCRIPT
  VARIABLE CACHE ON X ENABLE AS
  BEGIN
    x = SELECT * FROM test_cache;
    SELECT * FROM :x;
  END;
```

**Example - Use of READS SQL DATA with dynamic SQL** - The following examples show how to set dynamic SQL in the body of a procedure to read-only:

```
CREATE PROCEDURE Proc1() READS SQL DATA AS
  BEGIN
    EXEC 'SELECT * FROM DUMMY';
  END;

CREATE PROCEDURE Proc1(IN A NVARCHAR(12)) AS
  BEGIN
    EXECUTE IMMEDIATE 'SELECT * FROM ' || :A READS SQL DATA;
  END;
```

**Related Information**

- CREATE PROCEDURE
- ALTER PROCEDURE
- SAP HANA SQLScript Reference
- SAP HANA R Integration Guide
- ALTER PROCEDURE Statement (Procedural) [page 463]
- Table Variable Type Definition
- Library Members
- User-Defined Libraries
- CREATE LIBRARY Statement (SQLScript) [page 770]
- OBJECT_DEPENDENCIES System View [page 1604]
- M_SQLSCRIPT_VARIABLE_CACHE System view [page 2142]
4.10.1.61 CREATE PROJECTION VIEW Statement (Data Definition)

Creates a projection view. Projection views can be used as updatable views; insert, update, and delete triggers on projection views are supported.

Syntax

```
CREATE [ OR REPLACE ] PROJECTION VIEW <projection_view_name> [ ( <view_column_name_list> ) ] AS SELECT <table_column_list> FROM [ <schema>.]<source_table_name> [ <with_association_clause> ] [ <with_annotations_clause> ] [ WITH DDL ONLY ]
```

Syntax Elements

**OR REPLACE**

Replaces the projection view if it already exists; otherwise, if the view does not exist, then it is created.

A CREATE OR REPLACE operation does not change the ID of an existing projection view, and maintains any privileges associated with the view.

**projection_view_name**

Specifies a name for the new projection view.

**view_column_name_list**

Specifies alternative names for the columns in the projection view if the names need to be different than those in `<table_column_list>`.

**table_column_list**

Specifies the columns in `<source_table_name>` to include in the projection view. `<table_column_list>` must include all primary key columns of the underlying source table.

**source_table_name**

Specifies the source table for the projection view.

**with_association_clause**

Creates a relationship between the view being created and one or more existing tables.

```
<with_association_clause> ::= WITH ASSOCIATIONS (<association_def_list>) <association_def_list> ::= <association_def>, ... <association_def> ::= <forward_join_def> | <propagation_def> <forward_join_def> ::= <join_cardinality_class> [<<join_cardinality_class>]] ::= JOIN <table_or_view_identifier> [AS <table_alias>] ON <condition> | <propagation_def>
```
<join_cardinality> ::=   MANY TO ONE
                  | MANY TO MANY
                  | ONE TO ONE
                  | ONE TO MANY
<propagation_def> ::= [[<schema>.][<table>.]<association_identifier> [AS <alias>]]

WITH annotations

Specifies view-, column-, and parameter-level annotations in the form of key/value pairs. You can reference annotations in subsequent queries to filter results.

While you must specify annotation on at least one object with the WITH ANNOTATION clause (view, column, or parameter), there is no limit on the number of annotations or types of annotations you can specify.

<key> and <value> represent the annotations you are configuring for the object (view, column, or parameter).

WITH DDL ONLY

Prevents users from querying the view and from modifying the underlying table.

Description

You drop a projection view by using the normal DROP VIEW statement.

Examples

The following example creates a projection view on a table, SEARCH_TEXT, and then inserts, modifies, and deletes data from the projection view:

CREATE ROW TABLE SEARCH_TEXT( Product NVARCHAR(200) PRIMARY KEY, Descrip NVARCHAR(200), Status NVARCHAR(200) );
INSERT INTO SEARCH_TEXT VALUES( 'Blue baseball cap', 'Vintage', 'Out of stock' );
INSERT INTO SEARCH_TEXT VALUES( 'Red car', 'Vintage', 'Taking orders' );
CREATE PROJECTION VIEW myProjView (Item, Description, Status) AS SELECT * FROM SEARCH_TEXT;
INSERT INTO myProjView VALUES( 'Bluish sky', 'Retro', 'Discontinued' );
UPDATE myProjView SET Status = 'Discontinued' WHERE Item = 'Blue baseball cap';
DELETE FROM myProjView WHERE Item = 'Red car';
SELECT * FROM myProjView;

<table>
<thead>
<tr>
<th>ITEM</th>
<th>DESCRIPTION</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluish baseball cap</td>
<td>Vintage</td>
<td>Discontinued</td>
</tr>
<tr>
<td>Bluish sky</td>
<td>Retro</td>
<td>Discontinued</td>
</tr>
</tbody>
</table>

The following example creates two tables and a projection view with associations to them:

```
CREATE ROW TABLE T1(A INT, B INT);
INSERT INTO T1 VALUES(1,1);
CREATE ROW TABLE T2(A INT, B INT);
INSERT INTO T2 VALUES(1,1);
INSERT INTO T2 VALUES(2,2);
CREATE PROJECTION VIEW V1 AS SELECT A, B FROM T1 WITH ASSOCIATIONS(JOIN T2 AS C ON C.A = A);
SELECT C.A, C.B FROM V1;
```

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The following example creates a table with annotations and then creates a projection view that also has annotations:

```
CREATE TABLE t1(c1 INT) WITH ANNOTATIONS(
    SET 't_k1' = 't_v1', 't_k2' = 't_v2'
) COLUMN c1 SET 'c_k1' = 'c_v1', 'c_k2' = 'c_v2';
CREATE PROJECTION VIEW v2 AS SELECT * FROM t1 WITH ANNOTATIONS(
    SET 'v_k1' = 'v_v1', 'v_k2' = 'v_v2'
) COLUMN c1 SET 'c_k1' = 'c_v1', 'c_k2' = 'c_v2';
```

The following statements show you how to use the WITH DDL ONLY clause to control whether users can query a view:

```
CREATE TABLE t1 (a int, b int);
CREATE PROJECTION VIEW v1 AS SELECT * FROM t1 WITH DDL ONLY;
SELECT * FROM v1; -- invalid view name: cannot select the view because it is in DDL-only mode
SELECT IS_DDL_ONLY from VIEWS where VIEW_NAME = 'V1'; -- returns TRUE
CREATE PROJECTION VIEW v2 AS SELECT * FROM v1; -- you can select from the view in a DDL statement.
```

**Related Information**

- Projection Views (.hdbprojectionview and .hdbprojectionviewconfig)
- DROP VIEW Statement (Data Definition) [page 988]
4.10.1.62 CREATE PSE Statement (System Management)

Creates a personal security environment (PSE) store.

Syntax

CREATE PSE <pse_name>

Syntax Elements

<pse_name>

Specifies a unique name for the PSE store to be created.

<pse_name> ::= <identifier>

Description

Creates a personal security environment (PSE) store.

Permissions

You must have the TRUST ADMIN privilege to execute this statement.

Examples

Create a PSE store with the name example_pse.

CREATE PSE example_pse;

Related Information

ALTER PSE Statement (System Management) [page 468]
SET PSE Statement (System Management) [page 1167]
4.10.1.63 CREATE PUBLIC KEY Statement (System Management)

Creates a public key for JWT validation.

Syntax

```
CREATE PUBLIC KEY [ <public_key_name> ] FROM <pem> [ KEY ID HINT '<hint>' ]
[ COMMENT '<comment>' ]
```

Syntax Elements

- **public_key_name**
  - Specifies a unique name, which can be used to add the key to a PSE. If not specified, the name is auto-generated using the schema: "_SYS_PUBLIC_KEY_" + <id>
- **pem**
  - Specifies the key format.
  ```
  <pem> ::= { <PKCS1> | <PKCS8> }
  <PKCS1> ::= ('-----BEGIN RSA PUBLIC KEY-----...-----END RSA PUBLIC KEY-----')
  <PKCS8> ::= ('-----BEGIN PUBLIC KEY-----...-----END PUBLIC KEY-----')
  ```
  For the PKCS8 format, only RSA keys are supported. EC keys are not supported.
- **hint**
  - Used during the user authentication (JWT validation) to identify the key, if the JWT token contained a "kid" claim in its header.

Description

Creates a public key for JWT validation.

The database does not check for duplicate public keys. To check if a public key already exists in the database, see the PUBLIC_KEYS system view.
Permissions

You must have the CERTIFICATE ADMIN privilege to execute this statement.

Examples

Create a public key with the name jwt_pubkey.

```
CREATE PUBLIC KEY jwt_pubkey FROM '-----BEGIN PUBLIC KEY-----...-----END PUBLIC
KEY-----' KEY ID HINT 'key1';
```

Related Information

PUBLIC_KEYS System View [page 1623]

4.10.1.64  CREATE REMOTE SOURCE Statement (Access Control)

Defines an external data source that can connect to the SAP HANA database.

Syntax

```
CREATE REMOTE SOURCE <remote_source_name> <adapter_clause> [ <credential_clause> ]
```

Syntax Elements

remote_source_name

Specifies the identifier of the remote source.

```
<remote_source_name> ::= <identifier>
```

adapter_clause

Specifies the adapter configuration.

```
<adapter_clause> ::= 
```
adapter_name

Specifies the corresponding adapter, and the type of access method, to be used by the SAP HANA database to access the data. Query the ADAPTERS system view for the list of adapter names you can specify.

<adapter_name> ::= <identifier>

configuration_file

Specifies the configuration file for the specified adapter.

connection_info_string

Specifies the connection parameters for a given adapter.

For an on-premise remote source:

<connection_info_string> ::= 
  { 'DSN=<DSN_entry_name>'
  <host_name>):50111
  | Driver=<ODBC_library>; ServerNode=<machine_name>:3<instance#>15
  } [, <failover_server_name>:3<failover_instance_number>15 ]
  [ ;sessionVariable:<session_variable_name>=? ]
  [ ;linkeddatabase_mode=optimized ]
  } '  }

For an SAP HANA service instance remote source:

<connection_info_string> ::= 
  { 'DSN=<DSN_entry_name>'
  <host_name>):50111
  | Driver=<ODBC_library>; ServerNode=<tenant_endpoint> } 
  [, <failover Tenant_endpoint> ]
  [ ;sslTrustStore="digi_certificate_string";encrypt=TRUE; ]
  [ ;sessionVariable:<session_variable_name>=? ]
  [ ;linkeddatabase_mode=optimized ]
  } '  }

The failover, session variable, and linked database parameters are only supported for SAP HANA remote sources.

credential_clause

Specifies the credential configuration.

<credential_clause> ::= WITH CREDENTIAL TYPE <credential_type> [ USING <credentials> ]

credential_type

Supported credential types:

<credential_type> ::= { 'KERBEROS' | 'PASSWORD' | 'JWT' }

- PASSWORD: requires a cleartext user name and password to be used as credential information
- KERBEROS: uses the existing Kerberos configuration
• JWT: uses JSON Web Token (JWT) single sign-on (SSO) for smart data access (SDA)

**USING credentials**

Specifies the user name and password for `<credential_type>` PASSWORD.

```
<credentials> ::= 'user=<user_name>;password=<password>'
```

**Permissions**

This statement requires the CREATE REMOTE SOURCE privilege.

**Description**

For more information on creating remote sources, see *Creating a Remote Source* in the SAP HANA Administration Guide.

**Example**

This example creates a remote source named HOSTA on the hadoop adapter. Details for the remote Hadoop configuration are passed via the `webhdfs.url` and `webhcat.url` key/value pairs. The credentials user `dbuser` and password `DBtest123` are passed to the remote system.

```
CREATE REMOTE SOURCE HOSTA ADAPTER hadoop
  WITH CREDENTIAL TYPE 'PASSWORD' USING 'user=dbuser;password=DBtest123';
```

This example creates a remote source named HOSTB on the HANA adapter. Details for the remote ODBC configuration are passed via the `libodbcHDB.so` driver. Kerberos is used for authentication.

```
CREATE REMOTE SOURCE HOSTB ADAPTER hanaodbc
  CONFIGURATION 'Driver=libodbcHDB.so;ServerNode=my_hanaserver:30115;'
  WITH CREDENTIAL TYPE 'KERBEROS';
```

This example create a HANA remote source name HOSTC. Failover on the remote source is enabled. The session variable `APPLICATIONUSER` is set and optimized mode is enabled for linked database.

```
CREATE REMOTE SOURCE HOSTC ADAPTER hanaodbc
  CONFIGURATION 'Driver=livodbcHDB.so;ServerNode=my_hanaserver:30115,my_failover_hanaserver:30215
                     ;sessionVariable:APPLICATIONUSER=?;linkeddatabase_mode=optimized'
  WITH CREDENTIAL TYPE 'PASSWORD' USING 'user=dbuser;password=DBtest123';
```
4.10.1.65 CREATE ROLE Statement (Access Control)

Creates a new role.

Syntax

```
CREATE ROLE <role_name> [ LDAP GROUP <ldap_group_list> ] [ NO GRANT TO CREATOR ]
```

Syntax Elements

*role_name*

Specifies the name of the role to be created with optional schema name.

```
<role_name> ::= [<schema_name>.]<identifier>  
<schema_name> ::= <unicode_name>
```

The name must not be identical to the name of an existing user or role. If no schema is specified, then a global role is created. If you specify a schema as part of the role name, then the role name is local to that schema only and does not collide with roles with the same name in other schemas or in the global namespace. Dropping a schema drops all roles contained in the namespace of that schema.

*ldap_group_list*

Specifies the Distinguished Name (DN) of one or more LDAP groups.

```
<ldap_group_list> ::= <ldap_group_name> [,..]  
<ldap_group_name> ::= <string_literal>
```

A local role mapped to an LDAP group can also be granted to another role or a user as before. Note that when dropping an LDAP-mapped role using DROP ROLE, in addition to revoking the role from users to whom it was granted, its mapping to the LDAP group is deleted.

**NO GRANT TO CREATOR**

Prevents the automatic granting of the role to the user creating the role.
Description

For a list of system roles provided in the SAP HANA database, see the GRANT statement. To see all roles, including custom roles that have been created using the CREATE ROLE statement, query the ROLES system view. To view which roles and privileges have actually been granted, query the GRANTED_ROLES and GRANTED_PRIVILEGES system views, respectively.

Permissions

Only database users with the ROLE ADMIN system privilege can create roles.

Example

Create a role with the name role_for_work_on_my_schema:

```
CREATE ROLE role_for_work_on_my_schema NO GRANT TO CREATOR;
```

Create a role and associated it with an LDAP group:

```
CREATE ROLE Securities_DBA
  LDAP GROUP
  'cn=Securities_DBA,OU=Application,OU=Groups,ou=DatabaseAdmins,cn=Users,o=largebank.com';
```

4.10.1.66 CREATE SAML PROVIDER Statement (Access Control)

Defines a SAML provider in the SAP HANA database.

Syntax

```
CREATE SAML PROVIDER <saml_provider_name> <subject_issuer_clause>
  [ <entityid_clause> ]
  [ <user_creation_clause> ]
  [ <case_clause> ]
  [ <enable_user.creation_clause> ]
```
Syntax Elements

`saml_provider_name`

Specifies the identifier of a SAML provider to be created.

```
saml_provider_name ::= <simple_identifier>
```

The SAML provider name must not already be defined.

`subject_issuer_clause`

Sets SAML identity provider information.

```
subject_issuer_clause ::= WITH SUBJECT <subject_distinguished_name> ISSUER <issuer_distinguished_name>
```

`subject_distinguished_name`

Specifies the subject name provided in the certificate of the SAML identity provider.

```
subject_distinguished_name ::= <string_literal>
```

`issuer_distinguished_name`

Specifies the issuer name provided in the certificate of the SAML identity provider.

```
issuer_distinguished_name ::= <string_literal>
```

`entityid_clause`

Specifies the entity ID of the SAML identity provider.

```
entityid_clause ::= ENTITY ID <string_literal>
```

`case_clause`

Specifies whether the user mapping is checked case sensitive (default) or insensitive.

```
case_clause ::= CASE { SENSITIVE | INSENSITIVE } IDENTITY
```

`enable_user_creation_clause`

Enables automatic user creation when using SAML authentication, and specifies type of the user to create - standard or restricted.

```
enable_user_creation_clause ::= ENABLE USER CREATION [ USER TYPE { STANDARD | RESTRICTED } ] [ USERGROUP <usergroup_name> ] [ LDAP AUTHORIZATION ]
```

When enabling automatic user creation, optionally specify the type of the new users to be created. By default, new users are created as STANDARD users. The user is created in the specified user group. The RESTRICTED user created using this clause is not automatically granted the PUBLIC role. For information on LDAP User Authentication, see LDAP User Authentication in the SAP HANA Cloud, SAP HANA Database Security Guide.
Description

A SAML provider is required to provide a SAML connection facility for users. The `<subject_distinguished_name>` and the `<issuer_distinguished_name>` correspond to the subject and issuer of the X.509 certificate used by the SAML identity provider to sign assertions. The syntax of these names can be found in ISO/IEC 9594-1.

A detailed description of the concepts of SAML can be found in Oasis SAML 2.0.

Permissions

Only database users with the USER ADMIN system privilege can create a SAML provider. To use the `<enable_user_creation_clause>` also requires the OPERATOR object privilege on the referenced USERGROUP.

Example

Create a SAML provider with the name ac_saml_provider in the database specifying the subject and issuer to belong to ACompany. The entity ID is entity1. The identity checked by the provider is case sensitive.

```sql
CREATE SAML PROVIDER ac_saml_provider
  WITH SUBJECT 'CN = wiki.detroit.ACompany.corp,OU = ACNet,O = ACompany,C = EN'
  ISSUER 'E = John.Do@acompany.com,CN = ACNetCA,OU = ACNet,O = ACompany,C = EN'
  ENTITY ID 'entity1' CASE SENSITIVE IDENTITY;
```

Related Information

SAML_PROVIDERS System View [page 1648]
4.10.1.67 CREATE SCHEDULER JOB Statement (Data Definition)

Creates a scheduled job in the current or specified schema.

Syntax

```sql
CREATE SCHEDULER JOB [ <schema_name>.] <job_name>      CRON '<cron>' [ <job_recurrence_range> ]     [ ENABLE | DISABLE ] PROCEDURE [ <schema_name>.]<procedure_name> [ PARAMETERS <parameter_list> ]<NAME>=<CONST_EXPR> [, <NAME>=<CONST_EXPR>[, ...] ] )
```

Syntax Elements

- **job_name** Specifies the name of the scheduled job.
- **cron**

  Specifies the frequency of the job.

  ```sql
  <cron> ::= <year> <month> <date> <weekday> <hour> <minute> <seconds>
  
  year A four-digit number.
  month A number from 1 to 12.
  date A number from 1 to 31.
  weekday A three-character day of the week: mon,tue,wed,thu,fri,sat,sun.
  hour A number from 0 to 23 (expressed in 24-hour format).
  seconds A number from 0 to 59.
  
  Each cron field also supports wildcard characters as follows.
  
<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Any value.</td>
</tr>
</tbody>
</table>
  | */n       | Any n-th value. For example, */1 for the day of the month means run every day of the month, */2 every 2nd day.
  | a:b       | Any value between a and b. For example, 10:20 for the second specifies the range from 10:00 to 10:59.
  | a:b/n     | Any n-th value between a and b. For example, 1:10/3 for the day of the month means every 3rd day between 1 and 10 or the 3rd, 6th, and 9th day of the month.
  | n.a       | (For <weekday> only) A day of the week where n is a number from -5 to 5 for the n-th occurrence. For example, 2.3 means Tuesday, January 15th. -3.22 means Friday, May 31st.
  |
  
- **job_recurrence_range** Specifies the range for the recurrence of the scheduled job. If this value is not specified, then the recurrence of the scheduled job never ends.

  ```sql
  <job_recurrence_range> ::= FROM <start_time> ] [ UNTIL <end_time> ]
  ```
**start_time** Specifies the earliest time after which the scheduled job can start to run.

**end_time** Specifies the latest time before which the scheduled job can start to run.

**ENABLE | DISABLE** Specifies whether the scheduled job will run once the next scheduled time is reached. Use ALTER SCHEDULER JOB to enable a disabled scheduled job. If this clause is not specified, then the job is disabled by default.

**parameter_list**

Specifies the parameter for the specified procedure, in a comma-separated list of `<name> = <constant_expression>`.

```plaintext
<parameter_list> ::= <parm_name>=<parm_expression> [,<parm_name>=<parm_expression> [...]]
<parm_expression> ::= <name> = <constant_expression>
```

All IN or INOUT parameters without a DEFAULT must be provided.

**Permissions**

This statement requires the CREATE ANY privilege in the schema where the scheduled job was created and the EXECUTE privilege on the procedure referenced by the scheduled job.

**Example**

This example creates the retention_job scheduler job that runs from Monday to Friday at 1:23:45 and calls the procedure RETENTION_PROCEDURE. The three * mean that the scheduled job can run on any date (year month day). The job is enabled and the RETENTION_PROCEDURE requires the DAYS parameter, which is set to 14 in this example.

```sql
CREATE SCHEDULER JOB RETENTION_JOB CRON "* * * mon,tue,wed,thu,fri 1 23 45" ENABLE PROCEDURE RETENTION_PROCEDURE PARAMETERS DAYS=14;
```

### 4.10.1.68 CREATE SCHEMA Statement (Data Definition)

Creates a schema in the current database.

**Syntax**

```sql
CREATE SCHEMA <schema_name> [OWNED BY <user_name>]
```
Syntax Elements

**schema_name**
Specifies the schema name.

```plaintext
<schema_name> ::= <unicode_name>
```

**user_name**
Specifies the name of the schema owner. If omitted, the current user is the owner of the schema.

```plaintext
<user_name> ::= <unicode_name>
```

Description

The CREATE SCHEMA statement creates a schema in the current database.

Permissions

This statement requires the CREATE SCHEMA privilege.

Example

```
CREATE SCHEMA my_schema OWNED BY system;
```

4.10.1.69 CREATE SCHEMA SYNONYM Statement (Data Definition)

Creates a schema synonym.

Syntax

```
CREATE [OR REPLACE] SCHEMA SYNONYM <synonym_name> FOR <schema_reference>
```
Syntax Elements

synonym_name
The name of the synonym.
schema_reference
The name of the schema the synonym is for.

Description

The CREATE SCHEMA SYNONYM statement creates a synonym for an existing schema. Use the synonym when executing the SET SCHEMA statement, and, at connection time, the schema is redirected to the schema referenced by the synonym.

A schema synonym can be used in the following object definitions:

• Projection views
• SQL views
• Table functions
• Synonyms
• Procedures

Schema synonyms have the following restrictions:

• Synonyms cannot be used in the name of a new object:

```
CREATE TABLE SCHEMA_SYNONYM.TAB (A INT); -- error
```

• Synonyms cannot be created for the PUBLIC, SYS, or _SYS* schemas

• Synonyms cannot be created for the DEFAULT SCHEMA of procedures

• Synonyms cannot be used to update the current schema in SET SCHEMA:

```
CREATE SCHEMA SYNONYM SCHEMA_SYNONYM FOR SCHEMA_A;
SET SCHEMA SCHEMA_SYNONYM; -- current schema becomes SCHEMA_A
CREATE OR REPLACE SCHEMA SYNONYM SCHEMA_SYNONYM FOR SCHEMA_B; -- current schema will remain SCHEMA_A
```

• Synonyms cannot be used to refer to objects to be imported or exported:

```
EXPORT SCHEMA_SYNONYM."*" AS BINARY INTO '/tmp'; -- error
IMPORT SCHEMA_SYNONYM."*" AS BINARY FROM '/tmp'; -- error
IMPORT ALL AS BINARY FROM '/tmp' WITH RENAME SCHEMA 'SOURCE' TO 'SCHEMA_SYNONYM'; -- error
EXPORT INTO CSV FILE '/tmp/data.csv' FROM SCHEMA_SYNONYM.TBL; -- error
IMPORT FROM CSV FILE '/tmp/data.csv' INTO SCHEMA_SYNONYM.TBL; -- error
```

• Tables with associations which refer to schema synonyms cannot be imported:

```
IMPORT "SCHEMA"."TABLE_WITH_SCH_SYN_IN_ASSOC_CLAUSE" AS BINARY FROM '/tmp'; -- error
```

The synonym name cannot be the same as an existing schema name.
Permissions

You must have the CREATE SCHEMA system privilege.

Example

Execute the following statement to create a schema synonym called TEST_SCHEMA_SYN that references the schema SYSTEM:

```
CREATE SCHEMA SYNONYM TEST_SCHEMA_SYN FOR SYSTEM;
```

Execute the next statement to set the schema to the new schema synonym at connection time:

```
SET SCHEMA TEST_SCHEMA_SYN;
```

Now, when you execute `SELECT CURRENT_SCHEMA FROM DUMMY;`, it returns SYSTEM.

Execute the following statement to change the target schema from the schema synonym TEST_SCHEMA_SYN to the actual schema SYSTEM:

```
SET SCHEMA SYSTEM;
```

If you execute `SELECT CURRENT_SCHEMA FROM DUMMY;` again, it returns SYSTEM.

Related Information

- SET SCHEMA Statement (Session Management) [page 1170]
- DROP SCHEMA Statement (Data Definition) [page 971]
- DROP SCHEMA SYNONYM Statement (Data Definition) [page 972]
- CREATE SCHEMA Statement (Data Definition) [page 807]

4.10.1.70 CREATE SEQUENCE Statement (Data Definition)

Creates a sequence that generates primary key values that are unique across multiple tables, and for generating default values for a table.

Syntax

```
CREATE SEQUENCE <sequence_name> [ [sequence_parameter_list] ]
[ RESET BY <subquery> ]
```
Syntax Elements

sequence_name
Specifies the name of the sequence to be created, with optional schema name.

<sequence_name> ::= [ <schema_name>.]<identifier>

sequence_parameter_list
Defines one or more sequence parameters.

<sequence_parameter_list> ::= <sequence_parameter> [, <sequence_parameter> [, … ]] 

sequence_parameter
Specifies the sequence parameters.

<sequence_parameter> ::= START WITH <start_value> | INCREMENT BY <increment_value> | MAXVALUE <max_value> | NO MAXVALUE | MINVALUE <min_value> | NO MINVALUE | CYCLE | NO CYCLE | CACHE <cache_size> | NO CACHE

START WITH <start_value>
Specifies the starting sequence value. This can be any value from -9999999999999999999999999999 to 9999999999999999999999999999 (28 digits) if MINVALUE and/or MAXVALUE is extended. Otherwise, the START WITH value must be within the default MINVALUE and MAXVALUE range, which is 2^-63 to 2^63-1.

<start_value> ::= <signed_integer>

If you do not specify a value for the START WITH clause, then MINVALUE is used for ascending sequences and MAXVALUE is used for descending sequences.

INCREMENT BY <increment_value>
Specifies the amount that the next sequence value is incremented from the last value assigned.

<increment_value> ::= <signed_integer>

<increment_value> must be an integer between -9999999999999999999999999999 to 9999999999999999999999999999 (28 digits) if MINVALUE and/or MAXVALUE is extended. Otherwise, the INCREMENT BY value must be between the default MINVALUE and MAXVALUE range, which is 2^-63 to 2^63-1. The default is 1.

Specify a negative value to generate a descending sequence.

An error is returned if the INCREMENT BY value is 0.

MAXVALUE max_value
Specifies the maximum value that can be generated by the sequence.

```
MAXVALUE <max_value>
```

<max_value> must be a value between -9999999999999999999999999999 and 9999999999999999999999999999 (28 digits). If MAXVALUE is not specified the default MAXVALUE is set as described in the NO MAXVALUE directive.

**NO MAXVALUE**

When the NO MAXVALUE directive is used, the maximum value for an ascending sequence is $2^{63}-1$ and the maximum value for a descending sequence is -1.

**MINVALUE min_value**

Specifies the minimum value that a sequence can generate.

```
MINVALUE <min_value>
```

<min_value> must be a value between -9999999999999999999999999999 and 9999999999999999999999999999 (28 Digits). If MINVALUE is not specified the default MINVALUE is set as described in the NO MINVALUE directive.

**NO MINVALUE**

When the NO MINVALUE directive is used, the minimum value for an ascending sequence is 1 and the minimum value for a descending sequence is $-2^{63}$.

**CYCLE**

When the CYCLE directive is used, the sequence number restarts after it reaches its maximum or minimum value.

**NO CYCLE**

Specifies the default option. When the NO CYCLE directive is used, the sequence number does not restart after it reaches its maximum or minimum value.

**CACHE cache_size**

Specifies the cache size for caching sequence numbers in a node.

```
CACHE <cache_size>
```

<cache_size> must be an unsigned integer and cannot exceed $2^{31}-1$.

**NO CACHE**

Specifies to not cache sequence numbers in a node. This is the default behavior.

**RESET BY subquery**

During the restart of the database, the database automatically executes the `<subquery>` and the sequence value is restarted with the returned value.

If RESET BY is not specified, then the sequence value is stored persistently in database. During the restart of the database, the next value of the sequence is generated from the saved sequence value.

For more information on subqueries, see the SELECT statement.
Description

A sequence generates unique integers for use by multiple users. Sequence values are stored as DECIMAL.

Sequences do not support rollbacks. If a transaction is rolled back, the value of the sequence stays at the latest evaluated value. This is also the behavior for exceptions during fetching.

If MAXVALUE > 2^63-1(INT64_MAX) or MINVALUE < -2^63(INT64_MAX), then the return type of NEXTVAL or CURRVAL is DECIMAL type; otherwise, the return type is BIGINT. You can check the current value of a sequence by querying the M_SEQUENCES system view.

In the case of an ascending sequence (\(<increment_value> > 0\) ), the default MAXVALUE is 2^63-1, and the default MINVALUE is 1.

In the case of a descending sequence (\(<increment_value> < 0\) ), the default MAXVALUE is -1 and the default MINVALUE is -2^63.

Use CURRVAL (for example `SELECT <sequence_name>.CURRVAL FROM ...`) to get the current value of the sequence and NEXTVAL to get the next value of the sequence. CURRVAL is only valid after calling NEXTVAL in a session.

You can use CURRVAL and NEXTVAL only in:

- The select list of a SELECT statement which is not contained in a subquery, or view
- The select list of a subquery in an INSERT statement
- The VALUES clause of an INSERT statement
- The SET clause of an UPDATE statement

You cannot use CURRVAL and NEXTVAL in:

- The WHERE clause of a SELECT statement
- A subquery in a DELETE, SELECT, UPDATE, REPLACE or UPSERT statement
- A SELECT statement in a CREATE VIEW statement
- A SELECT statement with the DISTINCT operator
- A SELECT statement with a GROUP BY clause
- A SELECT statement with the UNION, INTERSECT, or MINUS set operator
- The DEFAULT value of a column in a CREATE TABLE or ALTER TABLE statement
- An Active/Active (read enabled) secondary system

Example

The following example creates a sequence `mySequence`, starting it at 1000, and then queries the NEXTVAL value for the sequence. Note that after you create a sequence, you cannot query the value of CURRVAL until the value is populated by querying the NEXTVAL value.

```sql
CREATE SEQUENCE mySequence START WITH 1000 INCREMENT BY 1;
SELECT mySequence.NEXTVAL FROM DUMMY;
```
Execute the following statement to get the value of CURRVAL:

```sql
SELECT mySequence.CURRVAL FROM DUMMY;
```

Now create a table, `myTable` with a column definition that references `mySequence.NEXTVAL`, and then query the contents of the table:

```sql
CREATE ROW TABLE myTable (a INT);
INSERT INTO myTable VALUES (mySequence.NEXTVAL);
SELECT * FROM myTable;
```

Now insert another row into `myTable` and query the contents of the table again:

```sql
INSERT INTO myTable VALUES (mySequence.NEXTVAL);
SELECT * FROM myTable;
```

The following example returns an error for the last statement because the sequence has been exhausted due to the default MAXVALUE. This error can be avoided by declaring a higher MAXVALUE when creating the sequence, as shown in the second CREATE SEQUENCE statement example.

```sql
CREATE SEQUENCE s1 START WITH 9223372036854775806;
SELECT s1.nextval FROM dummy; → the value 9223372036854775806 is returned
SELECT s1.nextval FROM dummy; → the value 9223372036854775807 is returned
SELECT s1.nextval FROM dummy; → an error is returned because the sequence is exhausted
```

```sql
DROP SEQUENCE S1;
CREATE SEQUENCE s1 START WITH 9223372036854775806 MAXVALUE 9999999999999999999999999999999;
SELECT s1.nextval FROM dummy; → the value 9223372036854775806 is returned
SELECT s1.nextval FROM dummy; → the value 9223372036854775807 is returned
SELECT s1.nextval FROM dummy; → the value 9223372036854775808 is returned
```
4.10.1.71 CREATE STATISTICS Statement (Data Definition)

Creates data statistic objects that allow the query optimizer to make better decisions for query plans.

Syntax

```
CREATE STATISTICS [ <data_statistics_name> ] ON <data_sources>
    [ <data_statistics_type> ]
    [ <data_statistics_properties> ]
    [ <initial_refresh> ]
```

Syntax Element

data_statistics_name

Specifies a unique name for the data statistics object.

```
<data_statistics_name> ::= [ <schema_name> [.]<identifier>
```

<data_statistics_name> is only allowed when the result of the creation is a single data statistics object. The number of data statistics objects created by CREATE STATISTICS is determined by the combination of <data_statistics_type> and the number of columns specified in <data_sources>.

```
<schema_name> ::= <identifier>
```

<schema_name> must be the same as specified for data source.

data_sources

Specifies the data sources you want to create data statistics objects for.

```
<data_sources> ::= <table_name> [ ( [ <column_name>[,]<column_name>[,...] ] ) ]
```

For RECORD COUNT data statistics objects, you cannot specify columns as part of <data_sources>.

table_name
Specifies the table name you want to create statistics on.

```plaintext
<table_name> ::= [ <database_name>.]<schema_name>.]<identifier>
```

For linked database, `<database_name>` is the name of the remote source. For all other cases, `<database_name>` is the name of the database where the table is located.

column_name

Specifies the column for which the data statistics are defined.

```plaintext
<column_name> ::= <identifier>
```

If no `<column_name>` is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

data_statistics_type

Specifies the type of data statistics object to create.

```plaintext
<data_statistics_type> ::= TYPE <type_name>
?type_name> ::= HISTOGRAM | SIMPLE | TOPK | SKETCH | RECORD COUNT
```

A data source can have only one data statistics object of a certain type. For example, column A of table T can have one data statistics object of type HISTOGRAM and one of type SIMPLE. If the TYPE clause is not specified, then the default is HISTOGRAM. Some data statistic types may not be appropriate for a given data source.

HISTOGRAM

Creates a data statistics object that helps the query optimizer estimate the data distribution in a single-column data source. If you specify multiple columns in `<data_sources>`, then multiple data statistics objects (HISTOGRAM) are created—one per column specified.

SIMPLE

Creates a data statistics object that helps the query optimizer calculate basic statistics, such as min, max, null count, count, and distinct count for a single-column data source. If you specify multiple columns in `<data_sources>`, then multiple data statistics objects are created—one per column specified. When beneficial, the SQL optimizer maintains system SIMPLE data statistics objects automatically on column and row store tables only.

TOPK

Creates a data statistics object that helps the query optimizer identify the highest-frequency values in a table data source. If you specify multiple columns in `<data_sources>`, then multiple data statistics objects are created—one per column specified. When beneficial, the SQL optimizer maintains system TOPK data statistics objects automatically (column store only).

SKETCH

Creates a data statistics object that helps the query optimizer estimate the number of distinct values in the data source. A data statistics object is created for the specified `<table_name>((<column-name>,...))`, which approximates the number of distinct tuples in the projection of the table on the set of specified columns.
SAMPLE

Creates a sample of data from `<data_source>` that the SQL optimizer can use during optimization. When beneficial, the SQL optimizer generates system SAMPLE data statistics objects automatically on column and row store tables. However, this behavior can incur a cost to performance. You can avoid this cost by creating SAMPLE data statistics objects explicitly (in advance). Creating them explicitly is especially useful in situations where sampling live table data is expensive (for example, very large tables).

`<sample_size_modifier> ::= SAMPLE SIZE <unassigned_integer>` defines the sample size. SAMPLE SIZE `<n>` is optional; if it is not given, the default value is 1000.

RECORD COUNT

Creates a data statistics object that helps the query optimizer calculate the number of records (rows) in a table data source. The RECORD COUNT type is a table-wide statistic. You do not specify columns in `<data_sources>` when creating a RECORD COUNT data statistics object. When beneficial, the SQL optimizer maintains system RECORD COUNT data statistics objects automatically on column and row store tables.

data_statistics_properties

Specifies the properties of the data statistics object.

```
<data_statistics_properties> ::=    <data_statistics_properties>[,<data_statistics_properties>[,...] ] 

<data_statistics_property> ::=    REFRESH TYPE <refresh_type>     ENABLE <on_off>     BUCKETS <unsigned_integer>     QERROR <numeric_literal>     QTHETA <unsigned_integer>     { MEMORY <memory_bytes> | MEMORY_PERCENT <memory_percentage> }     ACCURACY <numeric_literal>     PREFIX BITS <unsigned_integer>     PERSISTENT <on_off>     VALID FOR <valid_for_list>     CONSTRAINT '<constraint_param>'
```

Restrictions to which properties apply to which statistic types are noted in the property descriptions.

REFRESH TYPE refresh_type

Specifies the strategy for the data statistics object.

```
<refresh_type> ::= ( AUTO | MANUAL | DEFAULT )
```

AUTO specifies that the data statistics object is refreshed automatically when underlying data changes. AUTO is only supported on column store, extended store, and multistore tables.

MANUAL specifies that the database statistics object is not refreshed until a rebuild is explicitly requested by a REFRESH STATISTICS statement.

DEFAULT specifies that the database server decides the best refresh strategy based on the data source. For example, for data statistics objects on column store data sources, the database server applies AUTO for the default.

REFRESH TYPE only affects data statistics objects that are enabled.

ENABLE on_off
Controls whether the optimizer uses the data statistics object.

```plaintext
<on_off> ::= { ON | OFF }
```

ENABLE ON enables the optimizer to see the data statistics object. The data statistics object must be populated with data for the optimizer to use it. ENABLE ON specified with NO INITIAL REFRESH returns an error.

ENABLE ON is the default behavior.

ENABLE OFF disable the use of the data statistics object by the optimizer and prevents the ability to refresh the data statistics object. Data statistics objects that are not enabled can still be dropped. To make a data statistics object with ENABLE OFF accessible to the optimizer, execute an ALTER STATISTICS...ENABLE ON statement.

```plaintext
BUCKETS unsigned_integer
```

The BUCKETS property is only for use with TYPE HISTOGRAM or TOPK. For HISTOGRAM, BUCKETS specifies the maximum number of data buckets in the HISTOGRAM. For TOPK, BUCKETS specifies the K value.

The default is automatically determined by the data statistics building algorithm in use.

If a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

For column store, extended store, and multistore tables only, if a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

```plaintext
QERROR numeric_literal
```

Specifies the Q error to use for the q-optimal HISTOGRAM. You can specify this parameter when TYPE is HISTOGRAM and CONSTRAINT is QOPTIMAL. The default is automatically determined by the HISTOGRAM algorithm in use.

```plaintext
QTHETA unsigned_integer
```

Specifies a lower bound on the frequencies for which a q error constraint is applied for a q-optimal HISTOGRAM. You can specify this parameter when TYPE is HISTOGRAM and CONSTRAINT is QOPTIMAL. The default is automatically determined by the HISTOGRAM algorithm in use.

```plaintext
MEMORY memory_bytes
```

Specifies the maximum amount of memory, in bytes, to use for QOPTIMAL HISTOGRAMS.

The MEMORY parameter limits the memory for QOPTIMAL HISTOGRAMS. MEMORY applies only to the QOPTIMAL HISTOGRAM algorithm. Small values for MEMORY may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

```plaintext
MEMORY PERCENT memory_percentage
```

Specifies the maximum amount of memory to use for the data statistics object, expressed as a percentage of the space used by the data source.
HISTOGRAMS can use a large amount of memory for some data sources. \(<memory\text{-}percentage>\) represents the maximum amount of memory that can be used for the data statistics object. For example, if a data source is a table column that uses 100 MB of memory, and \(<memory\text{-}percentage>\) is 5, then the data statistics object for this column can use, at most, 5 MB for its in-memory representation.

The default is automatically determined by the HISTOGRAM algorithm in use. Small values for MEMORY PERCENT may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

**PERSISTENT on_off**

Specifies whether data statistics object data persists in the storage of the table, and only applies to QOPTIMAL HISTOGRAMS on column store tables. The default is PERSISTENT ON.

\(<on\_off> ::= \{ \text{ON} \mid \text{OFF} \} \)

Other statistics types are always persistent.

**ACCURACY numeric\_literal**

Controls the time and space requirements to use for the SKETCH algorithms. This parameter can only be specified when TYPE is SKETCH and must be a number between 0 and 1, with larger values causing decreased time and space requirements but poorer SKETCH resolution. The default is 0.1.

**PREFIX BITS unsigned\_integer**

Controls the number of bits the SKETCH algorithms use when constructing the SKETCH statistics. Specify this parameter when TYPE is SKETCH. Its value is an integer between 0 and 63. The default is 8.

**VALID FOR valid_for_list**

Defines how the data statistics object may be used. The VALID FOR clause is only permitted with statistics type SIMPLE.

\(<valid\_for\_list> ::= <usage>[, <usage>[,...]] \)

\(<usage> ::= \{ \text{ESTIMATION} \mid \text{DATA DEPENDENCY} \} \)

ESTIMATION initializes the data statistics object for use by the optimizer to improve selectivity estimation. SIMPLE data statistics objects are initialized for estimation use by default.

DATA DEPENDENCY applies to partitioned column store and multistore tables only. It initializes the data statistics object to be used by features that require higher (or more) data consistency (including automatically refreshing and rebuilding when needed), such as the dynamic partition pruning feature. For more information about the dynamic partition pruning feature, including the types of columns that can have data statistics defined for dynamic partition pruning, see the SAP HANA Administration Guide.

**CONSTRAINT constraint_param**

Specifies constraints to use for the specified \(<data\_statistics\_type>\).

- For HISTOGRAM, \(<constraint\_param>\) specifies the mathematical constraint for the HISTOGRAM:

\(<constraint\_param> ::= \{ \text{QOPTIMAL} \mid \text{MAXDIFF} \} \)
QOPTIMAL HISTOGRAMS are for column store tables only. MAXDIFF HISTOGRAMS are only for row tables, virtual tables, extended tables, and extended partitions of multistore tables. The defaults are QOPTIMAL for column tables and MAXDIFF for other data sources. A non-default CONSTRAINT for HISTOGRAMS results in an error. HISTOGRAM sizing restrictions (BUCKETS, MEMORY, and MEMORY PERCENT) are applied per HISTOGRAM.

- For SKETCH, `<constraint_param>` specifies the algorithm to use to build the SKETCH. The default is LOGLOGCOUNTING; the remaining algorithms are for internal use.

```
<constraint_param> ::= KMINVAL
                      PCSA
                      LINEARCOUNTING
                      LOGCOUNTING
                      LOGLOGCOUNTING
                      SUPERLOGLOGCOUNTING
```

**initial_refresh**

Specifies whether to populate the data statistics object with data after creation.

```
<initial_refresh> ::= [ NO ] INITIAL REFRESH
```

**INITIAL REFRESH**

Creates the definition of the data statistics object and populates it with data. The default behavior is INITIAL REFRESH.

**NO INITIAL REFRESH**

Creates the definition of the data statistics object, but does not populate it with data.

Use NO INITIAL REFRESH when you want to change the underlying data before refreshing the data statistics object.

You cannot specify NO INITIAL REFRESH if ENABLE OFF is not specified.

**Permissions**

One of the following is true with regards to required permission:

- You own the object you are creating statistics on.
- You have the ALTER privilege on the object you are creating statistics on.
- For linked database, you have the LINKED DATABASE object level privilege on the remote source, regardless of who created the remote source.

**Description**

The CREATE STATISTICS statement creates data statistics objects that approximate the specified `<data_sources>`. The optimizer uses data statistics objects to estimate the properties of the data without directly accessing the data. Data statistics objects can be useful during query optimization and query execution.
Examples

The following example creates a MAXDIFF HISTOGRAM with a maximum of 100 buckets on, the column T(X), where T is a virtual table.

```
CREATE STATISTICS ON MYSYSTEM.T(X) TYPE HISTOGRAM BUCKETS 100;
```

This example creates two SIMPLE statistics on two columns of table T where T is a virtual table, one for each column. The end result is the same as executing two CREATE STATISTICS statements on table T, one for each column.

```
CREATE STATISTICS ON MYSYSTEM.T(X,Y) TYPE SIMPLE;
```

The following example creates a q-optimal HISTOGRAM with a maximum 1000 buckets for each partition of the table on the column T(X), where T is a partitioned column store table.

```
CREATE STATISTICS ON MYSYSTEM.T(X) TYPE HISTOGRAM BUCKETS 1000;
```

The following example creates a SKETCH called SKETCH_T_XY with a manual refresh type for each partition in the table on the columns T(X) and T(Y), where T is a partitioned column store table.

```
CREATE STATISTICS MYSYSTEM.SKETCH_T_XY ON MYSYSTEM.T(X,Y) TYPE SKETCH REFRESH TYPE MANUAL;
```

The following example creates RECORD COUNT statistics for the virtual table MY_VIRTUAL_TABLE.

```
CREATE STATISTICS ON MYSYSTEM.MY_VIRTUAL_TABLE TYPE RECORD COUNT;
```

The following example creates TOPK statistics on T(A) with 1500 most frequently observed values on the remote source REMOTE1 using linked database:

```
CREATE STATISTICS ON REMOTE1.MYSYSTEM.T(A) TYPE TOPK BUCKETS 1500;
```

The following example creates a SIMPLE data statistics object, MYSIMPLESTAT on a partitioned column store table MYSYSTEM.MY_TABLE and enables it for use with dynamic partition pruning:

```
CREATE STATISTICS MYSYSTEM.MYSIMPLESTAT ON MYSYSTEM.MYTABLE(A) TYPE SIMPLE VALID FOR DATA DEPENDENCY;
```

The following example creates SAMPLE statistics with a SAMPLE SIZE of 10000 for a big table MY_TABLE:

```
CREATE STATISTICS ON MYSYSTEM.MY_TABLE TYPE SAMPLE SAMPLE SIZE 10000;
```

Related Information

CREATE STATISTICS Statement (Data Definition) [page 815]
ALTER STATISTICS Statement (Data Definition) [page 484]
REFRESH STATISTICS Statement (Data Definition) [page 1082]
DROP STATISTICS Statement (Data Definition) [page 974]
M_DATA_STATISTICS System View [page 1851]
4.10.1.72 CREATE STRUCTURED PRIVILEGE Statement (Access Control)

Creates a structured (analytic) privilege.

Syntax

```sql
CREATE STRUCTURED PRIVILEGE <privilege_name> FOR <action>
ON <object_name_list> <filter_condition>
```

Syntax Elements

- **privilege_name**
  Specifies a name for the privilege.
  ```sql
  <privilege_name> ::= [<schema>.]<identifier>
  ```

- **action**
  ```sql
  <action> ::= SELECT
  ```

- **object_name_list**
  Specifies the views to be restricted by this analytic privilege.
  ```sql
  <object_name_list> ::= <object_name> [{, <object_name>}...]
  <object_name> ::= [<schema>.]<identifier>
  ```

- **filter_condition**
  Specifies a view registered for a SQL-based analytic privilege check.
  Restrictions the returned rows of the specified views for users who have been granted this privilege.
  ```sql
  <filter_condition> ::= <static_filter_condition> | <dynamic_filter_condition>
  <static_filter_condition> ::= WHERE <condition>
  <dynamic_filter_condition> ::= CONDITION PROVIDER <procedure_name>
  <procedure_name> ::= [<schema>.]<identifier>
  ```

- **<condition>** restricts the rows returned and can contain subqueries.
- **<procedure_name>** specifies the procedure providing the dynamic restriction.
Description

Creates an analytic privilege. Analytic privileges provide fine-grained control over which data a user can see within a view. Analytic privileges are also referred to as structured privileges.

The new structured privilege is automatically applied to the owner/creator of the privilege; however, the owner is not listed as a grantee of the privilege in the GRANTED_PRIVILEGES system view because they have the privilege applied to them by virtue of ownership.

Permissions

You must have the CREATE STRUCTURED PRIVILEGE or STRUCTUREDPRIVILEGE ADMIN privilege to execute this statement.

Related Information

Analytic Privileges
STRUCTURED_PRIVILEGES System View [page 1662]
ALTER STRUCTURED PRIVILEGE Statement (Access Control) [page 491]
DROP STRUCTURED PRIVILEGE Statement (Access Control) [page 978]

4.10.1.73 CREATE SYNONYM Statement (Data Definition)

Creates an alternate name for a table, view, procedure, or sequence.

Syntax

```
CREATE [ OR REPLACE ] [ PUBLIC ] SYNONYM <synonym_name> FOR
<synonym_source_object_name>
```

Syntax Elements

**OR REPLACE**

Replaces the definition of the synonym if it already exists. A CREATE OR REPLACE operation does not change the ID of the synonym, nor any privileges associated with it.

**synonym_name**
Specifies the name of the synonym to be created, with optional schema name.

\[
\text{<synonym_name>} \ ::= \ [\text{<schema_name>}.]\text{<identifier>}
\]

When you create a synonym for PUBLIC, you cannot specify a schema name (\text{<schema_name>}).

\text{synonym_source_object_name}

Specifies the object you are creating a synonym for.

\[
\text{<synonym_source_object_name>} \ ::= \ \\
\text{<table_name>}
| \text{<view_name>}
| \text{<procedure_name>}
\]

\[
\text{<table_name>} \ ::= \ [\ [\text{<database_name>}.]\text{<schema_name>}.]\text{<identifier>}
\]

\[
\text{<view_name>} \ ::= \ [\ [\text{<schema_name>}.]\text{<identifier>}
\]

\[
\text{<sequence_name>} \ ::= \ [\ [\text{<schema_name>}.]\text{<identifier>}
\]

\[
\text{<procedure_name>} \ ::= \ [\ [\text{<schema_name>}.]\text{<identifier>}
\]

\[
\text{<database_name>} \ ::= \ <\text{identifier>}
\]

\[
\text{<schema_name>} \ ::= \ <\text{unicode_name>}
\]

\[
\text{<identifier>}\]

\text{<database_name>} can only be applied to tables and views, not to sequences or procedures.

The \text{<database_name>} syntax can be used only in a system with multiple database containers where cross-database access is enabled for the given database name.

For linked database, \text{<database_name>} is the name of the remote source. \text{<identifier>} is the name of the table on the remote source.

### Description

Use a synonym to re-point functions and stored procedures to differing tables, views, or sequences without rewriting the function or stored procedure.

The optional PUBLIC element allows you to create a public synonym.

### Permissions

To create or replace a synonym requires one of the following:

- For synonyms owned by self:
  - No additional privilege is required.
- For synonyms owned by others:
  - To create a synonym, you need one of the following:
    - CREATE ANY schema level privilege granted on the synonym.
  - To replace an existing synonym, you need one of the following:
    - CREATE ANY and ALTER schema level privilege granted on the synonym.
  - Regardless of synonym ownership, you require the proper privilege on its base objects within the synonym that you don't own.
Example

Create table A.

```sql
CREATE ROW TABLE A (A INT PRIMARY KEY, B INT);
```

Create a synonym for table A, a_synonym.

```sql
CREATE SYNONYM a_synonym FOR A;
```

4.10.1.74 CREATE TABLE Statement (Data Definition)

Creates a base or temporary table. See the CREATE VIRTUAL TABLE statement for creating virtual tables.

Syntax

```sql
CREATE [ <table_type> ] { <table_name> | <replica_name> }<table_contents_source>
    [ <system_versioning_spec> ]
    [ <application_time_period_configuration> ]
    [ <bi_temporal_table_spec> ]
    [ <with_association_clause> ]
    [ <with_annotation_clause> ]
    [ <with_mask_clause> ]
    [ <logging_option> ]
    [ <auto_merge_option> ]
    [ <unload_priority_clause> ]
    [ <schema_flexibility_option> ]
    [ <partition_clause> ]
    [ <persistent_memory_spec_clause> ]
    [ <group_option_list> ]
    [ <location_clause> ]
    [ <replica_clause> ]
    [ <with_replica_list> ]
    [ <global_temporary_option> ]
    [ <series_clause> ]
    [ <unused_retention_period_option> ]
    [ <record_commit_timestamp_clause> ]
    [ COMMENT <comment_string> ]
    [ <numa_node_preference_clause> ]
    [ <load_unit> ]
```
Defines the type of table to create.

```
<table_type> ::=   { ROW | [ COLUMN ] } TABLE
    | HISTORY COLUMN TABLE
    | <temporary_table_type>
    | VIRTUAL TABLE
<temporary_table_type> ::=   GLOBAL TEMPORARY { [ ROW ] | COLUMN } TABLE
    | LOCAL TEMPORARY { [ ROW ] | COLUMN } TABLE
```

If neither ROW nor COLUMN is specified, the default ROW for temporary tables (local and global), and COLUMN for non temporary tables.

**ROW and COLUMN TABLE**

Creates a row or column table, respectively. If the majority of table access is through a large number of tables with only a few selected attributes, then use COLUMN-based storage. If the majority of table access involves selecting a few records with all attributes selected, then ROW-based storage is preferable. The SAP HANA database uses a combination of table types to enable storage and interpretation in both ROW and COLUMN forms.

**HISTORY COLUMN TABLE**

Creates a column table with a particular transaction session type called HISTORY. Tables with the session type HISTORY support time travel queries. Time travel queries are queries against historical states of the database.

**temporary_table_type**

A simple comparison of the temporary table types is provided in this table. To learn more information about each type, see the descriptions found after the table.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Global temp (column)</th>
<th>Global temp (row)</th>
<th>Local temp (column)</th>
<th>Local temp (row)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metadata shared across sessions</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Dropped after session terminates</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Allows add or drop of primary key</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Allows add or drop of identity column</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**GLOBAL TEMPORARY ROW TABLE**

Creates a global temporary row table and makes the table definition globally available while data is visible only to the current session. Metadata for a global temporary table persists and is shared across sessions. The data in a global temporary table is session-specific, meaning only the session owner of a global temporary table can insert, read, or truncate the data for the session. As well, data in a temporary table is automatically dropped when the session is terminated. You can drop a global temporary table, but only after you truncate the session-specific data.
Supported operations:
- Create with or without a primary key
- Add or drop a primary key
- Add or drop an identity column
- Add or drop unique constraint
- Rename table
- Add, alter, rename, or drop column
- Truncate
- Drop
- Create or drop a view on a global temporary table
- Create synonym
- Select
- Select into or insert
- Delete
- Update
- Upsert or replace

**GLOBAL TEMPORARY COLUMN TABLE**

Creates a global temporary column table. The table definition becomes globally available, yet data visibility is confined to the current session.

Metadata for a global temporary table persists and is shared across multiple sessions. The data within a global temporary table is session-specific; only the session owner can insert, read, or truncate the data for that session. Additionally, data in a temporary table is automatically purged when the session concludes. A global temporary table can be dropped, but only after truncating the session-specific data.

Supported operations:
- Create without a primary key
- Truncate
- Drop
- Create or drop a view on a global temporary column table
- Create synonym
- Select
- Select into or insert

**LOCAL TEMPORARY ROW TABLE**

Creates a local temporary row table, visible only to the current session.

Metadata exists for the duration of the session and is session specific, meaning only the session owner of the local temporary table can see it. The table is dropped at the end of the session.

Data in a local temporary row table is session specific, meaning only the session owner of the local temporary row table can insert, read, or truncate the data.

Primary keys are supported on LOCAL TEMPORARY ROW tables.

Supported operations:
• Create with or without a primary key
• Truncate
• Drop
• Select
• Select into or insert
• Delete
• Update
• Upsert or replace

LOCAL TEMPORARY COLUMN TABLE

Creates a local temporary column table, visible only to the current session.

Metadata exists for the duration of the session and is session-specific, meaning only the session owner of the local temporary column table can see. The table is dropped at the end of the session.

Data in a local temporary column table is session-specific, meaning only the session owner of the local temporary column table can insert, read, or truncate the data.

Supported operations:
• Create without a primary key
• Truncate
• Drop
• Select
• Select into or insert

VIRTUAL TABLE

Creates a virtual table based on an existing table/view inside a remote source.

**table_name**

Specifies a name for the table

```
<table_name> ::= [ <schema_name>.]<identifier>
```

**replica_name**

Specifies the schema and a name for the new replica. This clause is for use with creating a replica based on an existing source table (CREATE TABLE...LIKE). See `<like_table_clause>` [page 839].

```
<replica_name> ::= [ <replica_schema_name>.]<identifier>
```

**table_contents_source**

Specifies the source from which the table definition is derived.

```
<table_contents_source> ::= 
  ( <table_element>, ... ) [ <with_association_clause> ]
  | [ <like_table_clause> |
  | [ ( <column_name>, ... ) ] <as_subquery>
```

You can create a table in the following ways:
• Describing the table elements (defined in this section).
• Basing the new table on an existing table (CREATE TABLE LIKE). See \texttt{<like_table_clause>} [page 839]).
• From the result of a subquery (CREATE TABLE AS). See \texttt{<subquery>} [page 841]).

table_element
Defines the table columns with associated column or table constraints.

\[
\text{<table_element> ::= <column_definition> \[ <column_constraint> \] \<table_constraint>)}
\]

column_definition
Defines a table column.

\[
\text{<column_definition> ::= <column_name> \{ <data_type> | <lob_data_type> \} \[ <ddic_data_type> \]} \[ <default_value_clause> \]} \[ <clientside_encryption> \]} \[ <col_gen_as_expression> \]} \[ <col_gen_as_ident> \]} \[ <col_calculated_field> \]} \[ <schema_flexibility> \]} \[ <fuzzy_search_index> \]} \[ <fuzzy_search_mode> \]} \[ <persistent_memory_spec_clause> \]} \[ COMMENT <comment_string> \]} \[ <load_unit> \]} \[ <numa_node_preference_clause> \]}
\]

column_name
Specifies the column name.

\[
\text{<column_name> ::= <identifier>}
\]

data_type
Specifies the column data types.

\[
\text{<data_type> ::= ARRAY} \]
\[
\text{DATE} \]
\[
\text{TIME} \]
\[
\text{SECONDDATE} \]
\[
\text{TIMESTAMP} \]
\[
\text{TINYINT} \]
\[
\text{SMALLINT} \]
\[
\text{INT} \]
\[
\text{BIGINT} \]
\[
\text{SMALLDECIMAL} \]
\[
\text{REAL} \]
\[
\text{DOUBLE} \]
\[
\text{TEXT} \]
\[
\text{BIENTEXT} \]
\[
\text{VARCHAR} \[ (<unsigned_integer>) \] \]
\[
\text{NVARCHAR} \[ (<unsigned_integer>) \] \]
\[
\text{ALPHANUM} \[ (<unsigned_integer>) \] \]
\[
\text{SHORTTEXT} \[ (<unsigned_integer>) \] \]
\[
\text{VARBINARY} \[ (<unsigned_integer>) \] \]
\[
\text{DECIMAL} \[ (<unsigned_integer> [, <unsigned_integer> ]) \] \]
\[
\text{FLOAT} \[ (<unsigned_integer>) \] \]
\[
\text{BOOLEAN} \]
\]

For tables with time-selection partitioning, the data type for the time-selection column must be NVARCHAR(8).
lob_data_type

Specifies the LOB data type.

```plaintext
<lob_data_type> ::= <lob_type_name> [ MEMORY THRESHOLD <memory_threshold_value> ]
```

lob_type_name

Specifies the type of LOB.

```plaintext
<lob_type_name> ::= { BLOB | CLOB | NCLOB }
```

ddic_data_type

The available data types.

```plaintext
<ddic_data_type> ::= DDIC_ACCP | DDIC_ALNM | DDIC_CHAR | DDIC_CDAY | DDIC_CLNT | DDIC_CUKY |
| DDIC_CURR | DDIC_D16D |
| DDIC_D34D | DDIC_D16R | DDIC_D34R | DDIC_D16S | DDIC_D34S |
| DDIC_DATS | DDIC_DAY | DDIC_DEC |
| DDIC_FLTTP | DDIC_GUID | DDIC_INT1 | DDIC_INT2 | DDIC_INT4 |
| DDIC_INT8 | DDIC_LANG | DDIC_LCHR |
| DDIC_MIN | DDIC_MON | DDIC_LRAW | DDIC_NUMC | DDIC_PREC |
| DDIC_QUAN | DDIC_RAW | DDIC_RSTR |
| DDIC_SEC | DDIC_SRST | DDIC_SSTR | DDIC_STRG | DDIC_STXT |
| DDIC_TIMS | DDIC_UNIT | DDIC_UTCM |
| DDIC_UTCL | DDIC_UCTS | DDIC_TEXT | DDIC_VARC | DDIC_WEEK |
```

memory_threshold_value

Specifies whether LOB data should be stored in memory.

```plaintext
<memory_threshold_value> ::= { <unsigned_integer> | NULL }
```

- If `<memory_threshold_value>` is not provided, then a hybrid LOB with a memory threshold of 1000 is created by default.
- If `<memory_threshold_value>` is provided and its LOB size is bigger than the memory threshold value, then the LOB data is stored on disk.
- If `<memory_threshold_value>` is provided and its LOB size is equal or less than the memory threshold value, then the LOB data is stored in memory.
- If `<memory_threshold_value>` is NULL, then all LOB data is stored in memory.
- If `<memory_table_lobthreshold_value>` is 0, then all LOB data is stored in disk.

This type of index can only be used for column store tables.

default_value_clause

Specifies a value to be assigned to the column if an INSERT statement does not provide a value for the column.

```plaintext
<default_value_clause> ::= DEFAULT <default_value_exp>    
<default_value_exp> ::= NULL |
| <string_literal> |
| <signed_numeric_literal> | <unsigned_numeric_literal> |
| <datetime_value_function> |
<datetime_value_function> ::= CURRENT_DATE |
| CURRENT_TIME
```
clientside_encryption

Enables client-side encryption on a column.

```sql
CLIENTSIDE ENCRYPTION ON WITH <column_encryption_key_name> [ RANDOM | DETERMINISTIC ]
```

RANDOM and DETERMINISTIC describe the type of encryption used by the algorithm to encrypt the data. RANDOM ensures that a given cleartext value is never encrypted into the same ciphertext more than once. DETERMINISTIC ensures that a given cleartext value is always encrypted into the same ciphertext value. Deterministic encryption allows some operations such as equality comparisons on encrypted data, whereas non-deterministic encryption (RANDOM), while stronger, limits operations to basic inserts, updates, and fetches. Choose the type of encryption based on how the data is intended to be used. RANDOM is the default setting.

Client-side encrypted columns are represented physically as VARBINARY.

- `<column_encryption_key_name>` must be a valid column encryption key, and you must have the USAGE permission on `<column_encryption_key_name>` to specify it.
- `<column_encryption_key_name>` and the table whose column is being encrypted must exist in the same schema.

**col_gen_as_expression**

Specifies the expression to generate the column value at runtime.

```sql
<col_gen_as_expression> ::= GENERATED ALWAYS AS <expression>
```

**col_gen_as_ident**

Specifies an identity column. Only one can be specified per table.

Generated columns are only supported in column store table definitions, and can only reference base columns that are defined prior to them in the DDL statement (that is, to the left of them). You cannot define a default for a generated column, and they cannot be used as part of a primary key or unique constraint.

You cannot drop a base column that is referenced by a generated column.
There are two types of IDENTITY columns: GENERATED ALWAYS, and GENERATED BY DEFAULT:

**GENERATED ALWAYS**

A GENERATED ALWAYS AS IDENTITY column increments based solely on the sequence parameter settings \(<sequence_param_list>\). With the exception changes to the RESET BY definition, the sequence defined for a GENERATED ALWAYS IDENTITY column cannot be altered or reset after it has been set, and inserting a value into a GENERATED ALWAYS identity column fails and returns an error.

Non-deterministic functions are not allowed for columns defined as GENERATED ALWAYS.

**GENERATED BY DEFAULT**

A GENERATED BY DEFAULT IDENTITY column increments using the sequence parameter settings \(<sequence_param_list>\) only when no identity value is specified in the INSERT statement (that is, the identity column is left out of the insert column list). When a value is specified for the identity column during the INSERT, then the specified value is inserted (assuming the value does not already exist) and the following behavior occurs with respect to the value of the identity column:

- If the specified value is higher than the existing identity value with a positive increment, then this resets the current sequence value to an allowable sequence value that is equal to the specified value, or the next one lower than the specified value.
- If the specified value is lower than the existing value with a negative increment, then this resets the current sequence value to an allowable sequence value that is equal to the specified value, or the next one higher than the specified value.
- Otherwise, the current sequence value is not reset.

Identity columns are supported in global and local temporary tables, both for row store and column store tables. They can be added to, or dropped from, a table. However, they cannot be altered.

If the RESET BY \(<subquery>\) clause is specified, then during the restart of the database, the database automatically executes the subquery and restarts the sequence value with the returned value.

If the RESET BY \(<subquery>\) clause is omitted, an automatic subquery is generated to obtain the next value for the identity column, taking into account the specified \(<increment_value>\). When the \(<increment_value>\) is positive, the subquery is designed to retrieve a value greater than the maximum value of the column. On the other hand, when the \(<increment_value>\) is negative, the subquery is designed to retrieve a value lower than the minimum value of the column.

For more information on subqueries, see the SELECT statement.

For more information on sequences and sequence parameters, see the CREATE SEQUENCE statement.

col_calculated_field
Specifies the expression that replaces the column at query compilation time.

\[
<\text{col\_calculated\_field}> ::= \text{AS} \ <\text{expression}>
\]

- The data type of a calculated field column can be a default value.
- The calculated field column can reference not only base columns, but also other calculated field columns.
- The maximum of recursive calculated field is 16.
- Dropping or altering the base columns of a calculated field is not possible.

**schema\_flexibility**

Specifies whether the column is dynamic.

\[
<\text{schema\_flexibility}> ::= \{ \text{ENABLE} \ | \text{DISABLE} \} \ \text{SCHEMA FLEXIBILITY}
\]

- ENABLE changes the column into a dynamic column.
- DISABLE (default value) changes the column into a static column.

**fuzzy\_search\_index**

Turns a fuzzy search index on or off (the default).

\[
<\text{fuzzy\_search\_index}> ::= \text{FUZZY SEARCH INDEX} \{ \text{ON} \ | \text{OFF} \}
\]

**fuzzy\_search\_mode**

Sets the fuzzy search mode with the value of \(<\text{string\_literal}>\).

\[
<\text{fuzzy\_search\_mode}> ::= \text{FUZZY SEARCH MODE} \{ \text{string\_literal} \ | \text{NULL} \}
\]

If NULL is specified, then the fuzzy search mode is reset.

**column\_comment\_string**

Specifies a descriptive comment for the column.

\[
<\text{column\_comment\_string}> ::= \text{string\_literal}
\]

Specifying COMMENT \(<\text{comment\_string}>\) saves you from having to execute a separate COMMENT ON statement later.

**column\_constraint**

Specifies a constraint on a column.

\[
<\text{column\_constraint}> ::= \\
\text{NULL} \\
\text{NOT NULL} \\
\{ \text{HIDDEN} \ | \text{NOT HIDDEN} \} \\
\{ \text{CONSTRAINT} \ <\text{constraint\_name}> \} \{ \text{<unique\_specification> |} \}
\]

**NULL**

Allows NULL values in the column. If NULL is specified it is not considered a constraint, it represents that a column may contain a null value. The default is NULL.

**NOT NULL**

Prohibits NULL values in the column.
HIDDEN | NOT HIDDEN

Specifies whether to hide the column; if not specified, the default behavior is NOT HIDDEN. A hidden column is excluded from a SELECT * operation on a table. It is also excluded in an INSERT INTO..VALUES operation unless the column list specifically references it. A hidden column still appears in system views such as TABLE_COLUMNS, INDEX_VIEWS, PARTITIONS, and so on. In the TABLE_COLUMNS system view, the IS_HIDDEN column indicates whether a column is hidden. Hidden columns still appear when using smart data access and in federation with other databases.

unique_specification

Specifies a uniqueness constraint on a column. There are two uniqueness constraints you can set: UNIQUE and PRIMARY KEY.

<unique_specification> ::=   UNIQUE [ <rs_tree_type_index> | <cs_inverted_type_index> ]    | PRIMARY KEY [ <rs_tree_type_index> | <cs_inverted_type_index> ]

When the optional index type (<rs_tree_type_index> or <cs_inverted_type_index> ) is omitted, the database server chooses the appropriate type by considering the table type and the column data type.

'rs_' and 'cs_' at the beginning of the specification types indicate which type of table the specification corresponds to (row store and column store, respectively). If you mistakenly specify a row store index type and the table is a column store table, then the specification is ignored and the index is created as INVERTED VALUE (default). If you mistakenly specify a column store index type and the table is a row store table, an error is returned.

For column-store tables, you can control the default index type behavior using configuration settings in indexserver.ini. Specifically, cs_composite_primary_key_constraint_index_type controls the default for multi-column primary keys, while cs_composite_unique_constraint_index_type controls the default for unique indexes. See the SAP HANA Configuration Parameter Reference for more information.

UNIQUE

Specifies that the index must have unique values. The optional <rs_tree_type_index> specification is only applicable to row store tables.

A composite unique key enables the specification of multiple columns as a unique key. With a unique constraint, multiple rows cannot have the same value in the same column.

A UNIQUE column can contain multiple NULL values.

PRIMARY KEY

Specifies a primary key constraint, which is a combination of a NOT NULL constraint and a UNIQUE constraint.

rs_tree_type_index

Specifies the tree type for the index. Tree type specification is only applicable to row store tables.

<rs_tree_type_index> ::= { BTREE | CPBTREE }

CPBTREE
Specifies a CPB+-tree index for row store tables. CPB+-tree stands for Compressed Prefix B+-Tree, which is based on pkB-tree. CPB+-tree is a very small index because it uses ‘partial key’ that is only part of full key in index nodes. CPB+-tree shows better performance than B+-Tree for larger keys. Specify this type of tree for the following scenarios:

- for character string types
- for binary string types
- for decimal types
- when the constraint is a composite key
- when the constraint is a non-unique constraint

**BTREE**

Specifies a B+-tree index. B+-tree indexes maintain sorted data, which performs efficient insertion, deletion, and search of records. Specify this type of tree for scenarios not described for CPBTREE.

**cs_inverted_type_index**

Specifies the inverted index type for column store tables. Inverted index types are only applicable to column store tables.

```
<cs_inverted_type_index> ::= INVERTED { HASH | VALUE | INDIVIDUAL }
```

**INVERTED HASH**

INVERTED HASH should not be used as a composite type in cases where range queries or similar queries on the composite keys are a significant part of the workload. In these cases, use INVERTED VALUE instead.

**Note**

Non-unique INVERTED HASH indexes are deprecated. Only UNIQUE INVERTED HASH indexes are supported.

**INVERTED VALUE**

INVERTED VALUE is the default index type for column store tables.

**INVERTED INDIVIDUAL**

An INVERTED INDIVIDUAL index is a lightweight index type for column store tables with reduced memory footprint. The name INVERTED INDIVIDUAL reflects that internally only inverted indexes on individual columns are created (that is, no concat attributes are created). This type of index is only available for multi-column unique constraints, which may be defined as secondary unique index or primary key.

For `<references_specification>`, see References Specification [page 836].

**table_constraint**

The table constraint can be either a unique constraint, a referential constraint, or a check constraint.

```
<table_constraint> ::= 
[ CONSTRAINT <constraint_name> ] 
{ <unique_constraint_definition>
```
unique_constraint_definition
Specifies unique constraints.

\[ \text{unique_constraint_definition} ::= \text{unique_specification} \ ( \text{unique_column_name_list} ) \]

unique_column_name_list
Specifies the unique column name list of one or more column names.

\[ \text{unique_column_name_list} ::= \text{unique_column_name}[, \text{unique_column_name} [..] ] \]

unique_column_name
Specifies a column name identifier.

\[ \text{unique_column_name} ::= \text{identifier} \]

referential_constraint_definition
Specifies a referential constraint.

\[ \text{referential_constraint_definition} ::= \text{FOREIGN KEY} \ ( \text{referencing_column_name_list} ) \ \text{references_specification} \]

To ensure uniqueness, the target of a foreign key constraint must be either a full primary key or a unique column. Self-referencing foreign key constraints are supported.

referencing_column_name_list
Specifies the target column(s) of the foreign key constraint.

\[ \text{referencing_column_name_list} ::= \text{referencing_column_name}[, \text{referencing_column_name} ] \]

Specifies that a hash encodes the composite key in a condensed manner. This option allows for faster equality queries over the composite keys. [....] Specifies that a hash encodes the composite key\] in a condensed manner. This option allows for faster equality queries over the composite keys, as well as reduced memory requirements for storage of the composite key.

referencing_column_name
The identifier of a referencing column.

\[ \text{referencing_column_name} ::= \text{identifier} \]

references_specification
Specifies the referenced table, with optional column=name list and trigger action.

\[ \text{references_specification} ::= \text{REFERENCES} \ \text{referenced_table} \ [ \ ( \text{referenced_column_name_list} ) \ ] \ [ \text{referential_triggered_action} ] \ [ \text{constraint_enforcement} ] \ [ \text{constraint_check_time} ] \]

\[ \text{constraint_check_time} ::= \text{INITIALLY} \ \{ \text{IMMEDIATE} \ | \text{DEFERRED} \} \]
**referenced_table**

Specifies the identifier of a table to be referenced.

```
<referenced_table> ::= <identifier>
```

**referenced_column_name_list**

Specifies the referenced column name list, which can have one or more column names.

If `<referenced_column_name_list>` is specified, then there is a one-to-one correspondence between `<column_name>` of `<column_definition>` and `<referenced_column_name>`. If it is not specified, then there is a one-to-one correspondence between `<column_name>` of `<column_definition>` and the column name of the referenced table's primary key.

```
<referenced_column_name_list> ::= <referenced_column_name>[, <referenced_column_name> [,…] ]
```

**constraint_check_time**

Specifies when to check constraints. The default is INITIALLY IMMEDIATE.

- IMMEDIATE - referential constraints are checked immediately during statement execution. If a referential constraint is violated, then the statement fails.
- DEFERRED - referential constraints are checked at commit time. If a referential constraint is violated, then the transaction is rolled back. Also, if `<referential_triggered_action>` is set to something other than RESTRICT, then the referential constraint check on the parent (referencing) table is not deferred and instead is checked immediately.

**referential_triggered_action**

Specifies an update rule with an optional delete rule or a delete rule with an optional update rule. The order in which they are specified provides an order of precedence for execution.

```
<referential_triggered_action> ::= <update_rule> [ <delete_rule> ] | <delete_rule> [ <update_rule> ]
```

**update_rule**

Specifies the update rule.

```
<update_rule> ::= ON UPDATE <referential_action>
<referential_action> ::= CASCADE | RESTRICT | SET DEFAULT | SET NULL
```

The following UPDATE referential actions are possible:
<table>
<thead>
<tr>
<th>Action Name</th>
<th>Update Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTRICT</td>
<td>Any updates to a referenced table are prohibited if there are any matched records in the referencing table. This is the default action.</td>
</tr>
<tr>
<td>CASCADE</td>
<td>If a record is updated in the referenced table, then the corresponding records in the referencing table are also updated with the same values. Any updates to a referenced table are prohibited if there are any matched records in the referencing table. This is the default action.</td>
</tr>
<tr>
<td>SET NULL</td>
<td>If a record is updated in the referenced table, then the corresponding records in the referencing table are also updated with null values.</td>
</tr>
<tr>
<td>SET DEFAULT</td>
<td>If a record is updated in the referenced table, then the corresponding records in the referencing table are also updated with their default values.</td>
</tr>
</tbody>
</table>

### delete_rule

Specifies the delete rule.

```sql
<delete_rule> ::= ON DELETE <referential_action>
```

The following DELETE referential actions are possible:

<table>
<thead>
<tr>
<th>Action Name</th>
<th>Delete Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESTRICT</td>
<td>Any deletions to a referenced table are prohibited if there are any matched records in the referencing table. This is the default action.</td>
</tr>
<tr>
<td>CASCADE</td>
<td>If a record in the referenced table is deleted, then the corresponding records in the referencing table are also deleted. When you specify NOT ENFORCED, new or modified rows are not checked to determine whether they violate the foreign key constraint. If a record in the referenced table is deleted, then the corresponding records in the referencing table are also deleted.</td>
</tr>
<tr>
<td>SET NULL</td>
<td>If a record in the referenced table is deleted, then the corresponding records in the referencing table are set to null.</td>
</tr>
<tr>
<td>SET DEFAULT</td>
<td>If a record in the referenced table is deleted, then the corresponding records in the referencing table are set to their default values.</td>
</tr>
</tbody>
</table>
**check_constraint_definition**

```xml
<check_constraint_definition> ::= CHECK ( <search_condition> )
```

Specifies a predicate to use as a table check constraint. A table check constraint is satisfied if `<search_condition>` evaluates to true.

**constraint_enforcement**

Enable or disable constraint enforcement and validation.

```xml
<constraint_enforcement> ::= <constraint_name> <constraint_enforcement>
<constraint_enforcement> ::= <enforcement_option>
<constraint_enforcement> ::= <validation_option>
<enforcement_option> ::= [NOT] ENFORCED
<validation_option> ::= [NOT] VALIDATED
```

When you specify NOT VALIDATED, existing data is not checked to determine whether it violates the foreign key constraint.

Enable or disable constraint enforcement rather than dropping and recreating the constraint.

The default `<enforcement_option>` is ENFORCED and the default `<validation_option>` is VALIDATED.

**like_table_clause**

Specifies a source table to be used for the definition.

```xml
<like_table_clause> ::= LIKE { <like_table_name> <table_creation_syntax> }
<like_table_clause> ::= [ WITH [ NO ] DATA ] [ <like_without_option> ]
<like_table_clause> ::= [ WITH INDEX ] [ <defined_index_name_list> ]
```

With the exception of mask definitions, all the column definitions with constraints, default values, and other properties, such as generated columns, schema flexibility, and so on, are copied from the table `<like_table_name>`. All table constraints, indexes, fulltext indexes, and the table location are also copied. Use the ALTER TABLE statement to add mask definitions to the new table.

**like_table_name**

Specifies the table being used for source definition information.

```xml
<like_table_name> ::= [ [ <database_name> ]<source_schema_name>.]<source_table_name>
<like_table_name> ::= <identifier>
```

Schema name and database name are optional. `<database_name>` is only for use in a multitenant system where cross-database access is enabled for the given database name.

**WITH [NO] DATA**

Specifies that data is copied from the `<like_table_name>` table. The default value of WITH NO DATA does not copy the table data.

**like_without_option**
Specifies which properties are not copied from the `<like_table_name>` table. For example, when WITHOUT PARTITION is specified, the table is not partitioned. The WITHOUT clause

```
<like_without_option> ::= 
  WITHOUT AUTO MERGE
  WITHOUT HISTORY
  WITHOUT NO LOGGING
  WITHOUT PARTITION
  WITHOUT SCHEMA FLEXIBILITY
  WITHOUT UNLOAD
  WITHOUT PRELOAD
  WITHOUT UNLOAD PRIORITY
  WITHOUT INDEX
  WITHOUT FUZZY SEARCH INDEX
  WITHOUT FUZZY SEARCH MODE
  WITHOUT GLOBAL TEMPORARY
  WITHOUT LOCAL TEMPORARY
  WITHOUT CONSTRAINT
  WITHOUT NUMA NODE
```

When you specify WITHOUT CONSTRAINT, the following restraints are not copied to the new table: PRIMARY KEY, UNIQUE, NOT NULL, CHECK constraints, and REFERENTIAL CONSTRAINT (foreign keys).

Use the WITHOUT NUMA NODE parameter to ignore NUMA NODE preferences that were set on the source table while creating the new table. If you do not include the WITHOUT NUMA NODE parameter, then SAP HANA compares the number of NUMA nodes on the target system with the number on the source system. If the number of NUMA nodes on the target system is greater than, or equal to, the number on the source system, then SAP HANA applies the node preference to the new table. If it has fewer NUMA nodes, then SAP HANA ignores the NUMA node preferences.

`defined_index_name_list` Specifies pairs of source and target index names. For each specified index on the source table, a matching index is created on the target table.

```
<defined_index_name_list> ::= <index_name_token> [, <index_name_token> ]
<index_name_token> ::= <source_index_name> AS <target_index_name>
```

`like_table_replica_specification` Creates a replica based on an existing table.

```
<like_table_replica_specification> ::= 
  [ <sync_or_async> ] REPLICA [ ( <src_table_column_list> ) ] 
  [ <replica_partition> ] 
  [ AT 'replica_indexserver_host:replica_indexserver_port' ] 
<br src_table_column_list> ::= <column_name> [, <column_name> [,...] ] 
<br replica_partition> ::= <partition_clause>
```

`replica_partition` Specifies the partitions for the replicated table.

```
<replica_partition> ::= <partition_clause>
```

The `<replica_partition>` clause supports all partition types not using time selection. See the definition for `<partition_clause>` later in this topic.

`as_subquery`
Creates a table and fills it with the data computed by the subquery. For more information about subqueries, see the SELECT statement topic.

```sql
<as_subquery> ::= { AS ( <subquery> ) [ WITH [ NO ] DATA ] | <with_clause> }
[ ADD <identity_column> ]

<with_clause> ::= { WITH <alias> AS ( <subquery> AS <alias><expression>
FROM <alias>

<identity_column> ::= ( <column_name> INT GENERATED ALWAYS AS IDENTITY( START WITH <integer> INCREMENT BY <integer> ) )
```

While both the AS <subquery> syntax and the [...] SELECT <with_clause> syntax create a table with data in it, the AS <subquery> syntax copies data from the columns derived from <subquery>, whereas the <with_clause> syntax allows you to define the data to insert into the columns.

If the <as_subquery> clause follows a <column_names> clause, then the column names specified in <column_names> override the column names defined in <as_subquery>.

If the table type is a history column, global temporary [column], or local temporary [column], then the memory threshold values of LOB columns are reset to NULL to store all LOB data in memory.

`<subquery>`

[... ] Defines the columns for the new table. If there are constraints on column derived from <subquery>, only the NOT NULL constraints are copied over.

WITH DATA inserts copies of the data from the columns in <subquery>; this is the default. If you specify WITH NO DATA, the table is created without any data.

`<with_clause>`

Defines the columns for the table as well as the data to insert into them.

ADD <identity_column>

Defines an ID column to add to columns derived from <subquery>. ADD <identity_column> syntax is not supported when creating virtual tables.

`<with_annotation_clause>`

Specifies table-, column-, and parameter-level annotations in the form of key/value pairs. You can reference annotations in subsequent queries to filter results.

```sql
<w ith_annotation_clause> ::= WITH ANNOTATIONS { { [ <set_table_annotations> ] [ <set_column_annotations> ] [ <set_parameter_annotations> ] } }<set_table_annotations> ::= <key_set_operation>
<set_column_annotations> ::= <column_annotation> [ <column_annotation> [...] ]
<column_annotation> ::= COLUMN <column_ref> <key_set_operation>
<set_parameter_annotations> ::= <parameter_annotation> [ <parameter_annotation> [...] ]
<parameter_annotation> ::= PARAMETER <column_ref> <key_set_operation>
<key_set_operation> ::= SET <key_value_pair_list>
<key_value_pair_list> ::= <key_value_pair> [ , <key_value_pair> [...] ]
<key_value_pair> ::= '<key>'=''<value>'
<column_ref> ::= <identifier>
<key> ::= <string>
=value> ::= <string>
```

While you must specify annotation on at least one object with the WITH ANNOTATION clause (table, column, or parameter), there is no limit on the number of annotations or types of annotations you can specify.
and \texttt{value} represent the annotations you are configuring for the object (table, column, or parameter).

\texttt{system\_versioning\_spec}

Configures system versioning for a table. System-versioning allows change tracking on database tables. This is achieved by adding validity-period columns to a table, and storing change information for the table in an associated history tables (not to be confused with HANA history tables).

\begin{verbatim}
\texttt{system\_versioning\_spec} ::= \{ \texttt{history\_table\_definition} | \texttt{system\_versioned\_table\_definition} \}
\texttt{history\_table\_definition} ::= ( \texttt{column\_definition} [,...] , \texttt{sys\_validfrom\_column\_name} TIMESTAMP NOT NULL, \texttt{sys\_validto\_column\_name} TIMESTAMP NOT NULL )
\texttt{system\_versioned\_table\_definition} ::= ( \texttt{column\_definition} [,...] , \texttt{sys\_validfrom\_column\_name} TIMESTAMP NOT NULL GENERATED ALWAYS AS ROW START, \texttt{sys\_validto\_column\_name} TIMESTAMP NOT NULL GENERATED ALWAYS AS ROW END, PERIOD FOR SYSTEM\_TIME( \texttt{sys\_validfrom\_column\_name}, \texttt{sys\_validto\_column\_name} ) [ WITH SYSTEM VERSIONING HISTORY TABLE \texttt{history\_table\_name} ] [ [ NOT ] VALIDATED ]
\texttt{column\_definition}

This is the normal data column definition as described in the \texttt{table\_elements} description and is only being included in the \texttt{history\_table\_definition} and \texttt{system\_versioned\_table\_definition} to demonstrate where to locate the system-versioning clauses relative to the overall table creation syntax.

\texttt{sys\_validfrom\_column\_name}

In both \texttt{history\_table\_definition} and \texttt{system\_versioned\_table\_definition}, this column defines the column that holds the start time of the validity period. Note that in \texttt{system\_versioned\_table\_definition}, an additional option \texttt{GENERATED ALWAYS AS ROW START} is added. This is the mechanism by which the system automatically updates the \texttt{TIMESTAMP} value in the column.

\texttt{sys\_validto\_column\_name}

In both \texttt{history\_table\_definition} and \texttt{system\_versioned\_table\_definition}, this column defines the column that holds the end time of the validity period. Note that in \texttt{system\_versioned\_table\_definition}, an additional option \texttt{GENERATED ALWAYS AS ROW END} is added. This is the mechanism by which the system automatically updates the \texttt{TIMESTAMP} value in the column.

\texttt{PERIOD FOR SYSTEM\_TIME( sys\_validfrom\_column\_name, valid\_to\_column\_name )}

Specifies the validity period columns, and is for multi-column unique constraints, which may be defined as secondary unique index or primary key.

\texttt{WITH SYSTEM VERSIONING HISTORY TABLE history\_table\_name}

Enables system-versioning for the table. \texttt{history\_table\_name} must already exist and not be associated with any other system-versioned table. If you create the table without this clause, system versioning is not enabled; you must subsequently execute an \texttt{ALTER TABLE} statement to add the clause and enable system versioning.
Specifies whether to check if `<history_table_name>` is empty. If VALIDATED is specified (the default), an exception is raised if `<history_table_name>` is not empty. If NOT VALIDATED is specified, no emptiness check is performed on `<history_table_name>`, and no exception is raised even if the history table has data in it.

**application_time_period_spec**

Creates an application-time period table. This option is only supported for column store tables.

```sql
<application_time_period_spec> ::= ( <column_definition_list>,   <validfrom_column_name> <date_type> NOT NULL,   <validto_column_name> <date_type> NOT NULL,   PERIOD FOR APPLICATION_TIME ( <validfrom_column_name>, <validto_column_name> ) )
```

**column_definition_list**

Specifies the column definition for the table. See the definition for `<column_definition>` earlier in this topic for more information about `<column_definition>`.

```sql
<column_definition_list> ::= <column_definition> [, <column_definition> [,..] ]
```

**validfrom_column_name and validto_column_name**

Specifies the names of the columns that will hold the period start and end times.

**datetime_type**

Specifies the datetime type of `<validfrom_column_name>` and `<validto_column_name>`.

**PERIOD FOR APPLICATION_TIME**

Specifies the validity period columns. The period adds an implicit constraint that `<validfrom_column_name>` must be less than `<validto_column_name>`.

**bi_temporal_table_spec**

Creates a bi-temporal table, which is a table that has both system-versioning and an application-time period defined on it. The syntax for creating a bi-temporal table combines the syntax elements from both types of table. The following syntax block shows this combination. However, see the `<system_versioning_spec>` and `<application_time_period_spec>` definitions in this topic for descriptions of the syntax elements.

```sql
<bi_temporal_table_spec> ::= { <bi_temporal_history_table_definition> | <bi_temporal_system_versioned_table_definition> }  
<bi_temporal_history_table_definition> ::= (    ... TIMESTAMP NOT NULL,    <validfrom_column_name> DATE NOT NULL,    <validto_column_name> DATE NOT NULL   )  
<bi_temporal_system_versioned_table_definition> ::= (     <column_definition_list>,    <sys_valid_from_column_name> TIMESTAMP NOT NULL GENERATED ALWAYS AS ROW START,     <sys_valid_to_column_name> TIMESTAMP NOT NULL GENERATED ALWAYS AS ROW END,     <validfrom_column_name> DATE NOT NULL,     <validto_column_name> DATE NOT NULL,
```
PERIOD FOR APPLICATION_TIME ( <validfrom_column_name>, <validto_column_name> ),
PERIOD FOR SYSTEM_TIME ( <sys_validfrom_column_name>, <sys_validto_column_name> ),
PRIMARY KEY ( <column_name> [, <column_name> [,..] ] )
[ WITH SYSTEM VERSIONING HISTORY TABLE <bi_temporal_history_table_name> ]

with_association_clause

Creates a relationship between the table being created and one or more existing tables.

<with_association_clause> ::= WITH ASSOCIATIONS ( <association_def_list> )
<association_def_list> ::= <association_def>, ...

association_def

Specifies the association relationship, including the join cardinality and DDL statement. Associations are checked at runtime, not creation time.

<association_def> ::= [
    [ <join_cardinality> ] JOIN <table_name> [ AS <identifier > ] ON <predicate> WITH DEFAULT FILTER <predicate>
    <join_cardinality> ::= MANY TO ONE
    | MANY TO MANY
    | ONE TO ONE
    | ONE TO MANY
    <table_name> ::= <identifier>

Use the WITH DEFAULT FILTER clause to specify a default predicate to filter column values.

with_mask_clause

Adds data masking to the contents of the specified columns. Data masking transforms confidential data so that it appears meaningless to users who don’t have the privileges required to view it.

<with_mask_clause> ::= WITH MASK ( <column_name> USING <mask_expression> [,..] )
<column_name> ::= <identifier>
<mask_expression> ::= <expression>

Masking behavior is supported on row and column tables, and SQL row and calculation views. It is not supported on any other type of tables (virtual tables, extended tables, temporary tables etc.) and views (Join/Olap/Hierarchy views).

Only one masking behavior, definer owner-based (DEFAULT MASK) or session user based masking (SESSION USER MASK), is supported on a table or view with masked columns. You can combine both masking behaviors in an object hierarchy. For example, a view with session user based masking can be created on a table with owner-based masking.

If masking is enforced on lower-level objects, results for higher-level objects, for unauthorized users, may be incorrect because some results data is masked; no error is returned indicating lower-level objects are masked.

If not specified, DEFAULT is the default.

<mask_expression> can be any type of expression, including a user-defined function, that returns the same data type and length as the original column.

For more information on data masking, see the SAP HANA Security Guide for SAP HANA Platform.
logging_option

Specifies logging options for the created table.

```plaintext
<logging_option> ::= LOGGING | NO LOGGING [ RETENTION <retention_period> ]
```

**LOGGING**

Activates table logging. This is the default logging option.

**NO LOGGING**

Deactivates logging. A NO LOGGING table means that the definition of the table is persistent and globally available with data that is temporary and global.

**RETENTION**

Specifies the retention time in seconds of the column table created by NO LOGGING. The table is temporary and globally available with data that is temporary and global.

```plaintext
RETENTION <retention_period>
<retention_period> ::= <unsigned_integer>
```

After the specified retention period elapses, the table is dropped if the used physical memory on the host is above 80%.

**i Note**

A NO LOGGING retention table can also be created on an Active/Active (read enabled) secondary system.

auto_merge_option

Specifies that an automatic delta merge is triggered; this is the default.

```plaintext
<auto_merge_option> ::= [NO] AUTO MERGE
```

NO AUTO MERGE disables automatic delta merging.

unload_priority_clause

Specifies the priority of the table to be unloaded from memory.

```plaintext
<unload_priority_clause> ::= UNLOAD PRIORITY <unload_priority>
```

`<unload_priority>` is a number from 0 and 9, inclusive, where 0 means not-unloadable and 9 means earliest unload.

schema_flexibility_option

Specifies that the table schema is flexible. The flexibility option is available for column tables.

```plaintext
<schema_flexibility_option> ::= WITH SCHEMA FLEXIBILITY [(<flexibility_option>...)]
<flexibility_option> ::= RECLAIM | DEFAULT DATA TYPE <default_data_type> | NO DEFAULT DATA TYPE | AUTO DATA TYPE PROMOTION | NO AUTO DATA TYPE PROMOTION
```
Dynamic columns in flexible tables allow you to implicitly add columns and change column data types with INSERT, UPDATE, and REPLACE/UPsert operations.

Dynamic columns can be created automatically during a data insertion operation, an update operation, or a replace/upsert operation. Spatial and text data types are not available in the dynamic schema of flexible tables. The deletion of automatically created columns is done asynchronously in the background through a dedicated garbage collection mechanism.

Implicit DDL operations during DML operations are auto-committed and cannot be rolled back.

**RECLAIM**
Enables garbage collection for a flexible table. If it is enabled, then dynamic columns are automatically dropped when no values are left for all rows in the column. The garbage collection runs periodically according to the interval specified as reclaim_interval in the flexible_table section of the indexserver.ini file. Trigger garbage collection manually with the ALTER SYSTEM RECLAIM FLEXIBLE TABLES statement.

**NO RECLAIM**
Turns off garbage collection.

**DEFAULT DATA TYPE**
Specifies the default data type.

```
DEFAULT DATA TYPE {<default_data_type> | *}
```

By default an NVARCHAR with a maximum length of 5000 is chosen as the data type for new dynamic columns in flexible tables. This default behavior of a flexible table can be changed by explicitly specifying a default data type. To let the database automatically detect and choose the best matching data type, specify * as a wildcard for the default data type.

Database automatic data type detection is only supported for constant values (expressions are not supported).

**NO DEFAULT DATA TYPE**
Explicitly forces the use of the system default for the default data type (NVARCHAR(5000)).

**AUTO DATA TYPE PROMOTION**
Alters the column type of a dynamic column to a more generic data type that is able to store the old and the new data without any loss of information when new data is inserted. Automatic data type promotion can be combined with setting a default data type.

**NO AUTO DATA TYPE PROMOTION**
Explicitly disables auto data type promotion.

**partition_clause**
For clauses for creating heterogeneous and non-heterogeneous partitions, see the topics on Non-heterogeneous Create Partition Clauses Heterogeneous Create Partition Clauses in this guide.

**Non-heterogeneous Create Partition Clauses**
**Heterogeneous Create Partition Clauses**

**persistent_memory_spec_clause**
Enables or disables persistent memory storage preference at the table, range partition, or column level, depending on where the clause is situated in the CREATE statement. For example, when specified inside the `<column_definition>` clause, it enables or disables persistent memory storage for the column.

```plaintext
<persistent_memory_spec_clause> ::= PERSISTENT MEMORY <pm_preference>
<pm_preference> ::= { ON | OFF }
```

**load_unit**

Specifies how to load data into memory when the table is queried. Specifying the load unit is only supported on column-store tables. `<load_unit>` can be set at column, table, and partition levels.

```plaintext
<load_unit> ::= { COLUMN | PAGE } LOADABLE
```

**COLUMN LOADABLE**

In-memory loading - the entire column is loaded into memory. COLUMN LOADABLE boosts performance at the cost of higher memory usage. This is the default behavior unless another value is inherited.

**PAGE LOADABLE**

In-buffer cache loading - column data is loaded by page into the buffer cache. PAGE LOADABLE reduces memory usage for specific columns by not requiring those columns to be fully memory resident.

In the case of competing or unspecified `<load_unit>` settings, the following logic is applied.

- If there is a load unit preference specified for a column, apply that value.
- Else if there is a load unit preference specified for the parent partition, apply that value.
- Else if there is a load unit specified for the parent table, apply that value.
- Else apply the default (COLUMN LOADABLE).

**group_option_list**

Specifies the group type, subtype, and name, and whether the table is the leader of the table group.

```plaintext
<group_option_list> ::= <group_option> [ <group_option> ...]
<group_option> ::= GROUP TYPE <identifier> |
GROUP SUBTYPE <identifier> |
GROUP NAME <identifier> [ GROUP LEAD ]
```

GROUP LEAD sets the table as the leader of the table group it belongs to.

**replica_clause**

Creates one or more table replicas at the same time the source table is created.

To create a column-wise replica that is based on an existing table (CREATE TABLE LIKE syntax), see the `<like_table_replica_specification>` syntax in this topic.

```plaintext
<replica_clause> ::= [ { sync_or_async } [ <column_or_row> ] REPLICA [ ( <src_table_column_list> ) ] [ <replica_partition> ] ] [ AT <replica_locations> ]
```

**sync_or_async**
Specifies when to activate and populate the replica. The default is SYNCHRONOUS, which means the replica is created, populated with data, and enabled when the source table is created.

\[
<\text{sync_or_async}> ::= \{ \text{SYNCHRONOUS} \mid \text{ASYNCHRONOUS} \}
\]

Replicas created as ASYNCHRONOUS are created empty and disabled; you must execute an ALTER SYSTEM ENABLE ALL ASYNCHRONOUS TABLE REPLICAS statement to enable them.

\textit{Column or row}

Specifies whether the replica is a column or row table. If \textit{<column_or_row>} is not specified, then the type is the same as that of the source table.

\[
<\text{column_or_row}> ::= \{ \text{COLUMN} \mid \text{ROW} \}
\]

\textit{Replica partition}

Specifies the partitions for the replicated table.

\[
<\text{replica_partition}> ::= <\text{partition_clause}>
\]

The \textit{<replica_partition>} clause supports all partition types not using time selection. See the definition for \textit{<partition_clause>} earlier in this topic.

\textit{Source table column list}

Specifies a subset of the columns in the source table. A replica that has a subset of the columns from the source table is called a \textit{column-wise replica}.

\[
<\text{src_column_list}> ::= <\text{src_column_name}> [, <\text{src_column_name}> [,…] ]
\]

Although you can specify the columns in any order, the order of the columns in the replica will match the order of columns in the source table.

\textit{AT replica locations}

Creates replicas at the specified locations.

\[
<\text{replica_locations}> ::= \text{AT} \{ <\text{indexserver_host_port}> \mid ( <\text{indexserver_host_port}> , \ldots ) \} <\text{indexserver_host_port}> ::= '\text{hostname}:<port_number>'
\]

If you do not specify a location, then the replica table is created based on the table placement rule. If no table placement rule is also defined, then the first replica table is created on one of slave nodes that is not the source table location.

The former syntax, AT \{ ALL LOCATIONS \mid <\text{num_replicas}> LOCATIONS \}, is supported but deprecated. Specify locations using the AT \textit{<replica_locations>} syntax instead.

\textit{With replica list}

Creates one or more table replicas at the same time the source table is created. Optionally, the replica tables can be grouped.

\[
<\text{with_replica_list}> ::= <\text{with_replica_clause}> [\ldots]
\]

\[
<\text{with_replica_clause}> ::= \text{WITH} \[ \{ <\text{sync_or_async}> \} \[ <\text{column_or_row}> \]
\text{REPLICA} \[ ( <\text{src_table_column_list}> ) \] \[ <\text{replica_partition}> \]
\[ <\text{set_group_option}> \] [\ \text{AT} <\text{replica_locations}> ]
\]
Specifies the index servers on which the partitions are created in a round-robin scheme.

\[
\text{<location_clause> ::= \{ [ NOT ] MOVABLE ] AT [LOCATION] (\'<hostname>:<port_number>\')}\]

A table can be created in the specified location with \(\text{<hostname>:<port_number>.<location_clause>}\) can be specified when creating partitioned tables that are distributed on multiple instances. When \(\text{<location_clause>}\) is provided without \(\text{<partition_clause>}\), the table is created on the first location specified. If location information is not provided, then the table is automatically assigned to one node. This option can be used for both row-store and column-store tables in a distributed environment.

Use the \([\text{NOT}]\) MOVABLE clause to control whether the table can be moved to another location. When a table is set to NOT MOVABLE, it cannot be moved to another location. By default, tables are movable.

**global_temporary_option**

Sets data availability of global temporary tables within session or transaction level. Only global temporary tables can use this option.

\[
\text{<global_temporary_option> ::= \{<global_temporary_preserve> \| <global_temporary_delete>\}}\]

**global_temporary_preserve**

Causes data to remain available within a session. The data is deleted when the session is terminated.

\[
\text{<global_temporary_preserve> ::= \{ON COMMIT PRESERVE ROWS\}}\]

PRESERVE is the default value. Therefore, global temporary tables use session data without this option.

**global_temporary_delete**

Causes data to remain available within a transaction. The data is deleted when the transaction is committed.

\[
\text{<global_temporary_delete> ::= \{ON COMMIT DELETE ROWS\}}\]

**series_clause**

Specifies settings for a table that contains series data.

\[
\text{<series_clause> ::= SERIES \{ [<series_key>] [<series_equidistant_definition>] [<series_minvalue>] [<series_maxvalue>] [ <series_period> [<alternate_series>]] \}}\]

**series_key**

Defines one or more columns that uniquely identify a particular series within the table. The table stores a single series when no series key columns are defined. For series tables that do not allow duplicate entries, a PRIMARY KEY can be defined with the SERIES_KEY columns and a period column.

\[
\text{<series_key> ::= SERIES KEY \{<series_key_column> [ {, <series_key_column> \_ \_ } ] \}}\]

**series_equidistant_definition**
Specifies whether the table is equidistant.

```sql
<series_equidistant_definition> ::= NOT EQUIDISTANT | EQUIDISTANT INCREMENT BY <increment_by> [ <missing_elements> ]
```

When using an equidistant series for table partitioning, use HASH or RANGE partitioning so that records with the same series key are in the same partition.

**increment_by**

Defines the distance between adjacent elements.

```sql
<increment_by> ::= <real_const> | <datetime_const>
```

The delta is defined by an integer constant, an interval constant, or a defined number of elements between a minimum and maximum. The delta is used in the physical storage representation to optimize an equidistant series table. For numeric series, the `increment_by` value defines the period boundaries with respect to numeric zero (0). For example, if the `increment_by` is 3, then valid period boundaries include 0, 3, 6, ... Define `increment_by` as a `real_const` or a `datetime_const`. It is either an integer constant, an interval constant, or a defined number of elements between a minimum and maximum, so it can be an integer or a real. If the period type is a DATETIME, then the number needs to be preceded by `INTERVAL` followed by a unit, such as `YEAR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, or `SECOND`. Exponential notation is allowed.

**missing_elements**

Indicates whether missing elements are allowed.

```sql
<missing_elements> ::= MISSING ELEMENTS [ NOT ] ALLOWED
```

Equidistant series define a mapping from the series data type to elements. A series table has no missing elements when there is a single contiguous range of elements where every element has a matching row. Otherwise, the table has missing elements.

**series_minvalue and series_maxvalue**

Specifies a minimum or maximum value.

```sql
<series_minvalue> ::= NO MINVALUE | MINVALUE <datetime_literal>
<series_maxvalue> ::= NO MAXVALUE | MAXVALUE <datetime_literal>
```

For a non-equidistant series, any representable value between the minimum and maximum values is permitted. When a minimum or maximum value is specified for an equidistant series, the value must be aligned with the `increment_by` value with respect to the natural zero. For example, a numeric series with an `increment_by` value of 2 requires the minimum and maximum value to be an even number. For date/time types with an `increment_by` unit of `DAY` or smaller, the zero is considered to be midnight. For a unit of `MONTH`, the zero is the first day of the month, and for a unit of `YEAR`, the zero is the first day of the year.

```sql
<constant_literal> ::= <real_const> | <datetime_const>
<interval_const> ::= INTERVAL <integer> <unit>
<unit> ::= YEAR | MONTH
```
**series_period**

Specifies the period of validity in each row that represents the period of time that the row applies to.

```sql
<series_period> ::= PERIOD FOR SERIES <series_period_columns>
<series_period_columns> ::=    (<series_instant_column>)   | (<series_period_start_column>, <series_period_end_column>)
```

There is a single period column when the series table represents instants. For intervals, there can be one or two columns.

**series_instant_column**

Represents the instant in time that the single period row applies to.

```sql
<series_instant_column> ::= <identifier>
```

**series_period_start_column and series_period_end_column**

Defines an interval series table.

```sql
<series_period_start_column> ::= <identifier> | NULL
<series_period_end_column> ::= <identifier> | NULL
```

The start and end of the interval are maintained separately. For an equidistant series, INCREMENT_BY allows the user to find either end point given the other, so you can omit either start or end from the series table without loss of information. In all cases for an interval, the interval is defined with closed-open semantics.

**alternate_series**

Represents one or more columns expected to be offset from the start time by an offset that changes infrequently, such as local-time columns reflecting changes in daylight savings time.

```sql
<alternate_series> ::= ALTERNATE PERIOD FOR SERIES (<column1> [, <column2> ... ] )
```

**unused_retention_period_option**

Defines the retention period for a table or table partitions in seconds.

```sql
<unused_retention_period_option><retention_period> | ( <retention_period>, ...) )
<retention_period> ::= <unsigned_integer>
```

A single value specifies the retention period at table-level, including all of the table partitions. Multiple values specify individual retention periods for each of the table partitions. If multiple values are specified, then the number of values must match the number of table partitions.

The default value and behavior of UNUSED_RETENTION_PERIOD is 0 (inactive) seconds. To activate UNUSED_RETENTION_PERIOD, change the value to something other than 0. However, use of the UNUSED_RETENTION_PERIOD setting on a table requires that the global unused_retention_period setting in the global.ini configuration file be set to something other than 0.

**record_commit_timestamp_clause**
Specifies that the COMMIT timestamp of a row is tracked. If this option is enabled, then every row is associated with the COMMIT timestamp of the last transaction that performed a DML operation on the row.

<record_commit_timestamp_clause> ::= RECORD COMMIT TIMESTAMP

You cannot use the RECORD COMMIT TIMESTAMP clause when creating the following tables:

- Temporary tables
- Virtual tables
- Multi-store tables
- Proxy tables
- History tables
- Asynchronous/synchronous replicated tables

Synchronous replicated tables and tables created with the RECORD COMMIT TIMESTAMP clause cannot be updated together or the COMMIT operation fails.

Depending on the table type, DDL statements, such as ALTER TABLE or CREATE TABLE LIKE, can affect the value returned by the RECORD_COMMIT_TIMESTAMP() function:

- For column store tables, the returned value is zero if the result of a CREATE TABLE LIKE statement, otherwise the value is unaffected by the DDL.
- For row store tables, the returned value is changed to the COMMIT timestamp of the transaction that performed the DDL operation.

You cannot use RECORD COMMIT TIMESTAMP with keys or indexes if you are creating a row store table.

For example, the following CREATE TABLE statements cannot be executed:

```sql
CREATE ROW TABLE ROWCTS_LIMIT (X INT PRIMARY KEY) RECORD COMMIT TIMESTAMP;
CREATE ROW TABLE ROWCTS_LIMIT (X INT UNIQUE) RECORD COMMIT TIMESTAMP;
```

The following ALTER TABLE statements cannot be executed for row store tables that have been created with the RECORD COMMIT TIMESTAMP clause:

```sql
ALTER TABLE ROWCTS_LIMIT ADD (B INT PRIMARY KEY);
ALTER TABLE ROWCTS_LIMIT ALTER (X INT PRIMARY KEY);
ALTER TABLE ROWCTS_LIMIT ADD CONSTRAINT UK UNIQUE (X);
ALTER TABLE ROWCTS_LIMIT ADD (c INT UNIQUE);
ALTER TABLE ROWCTS_LIMIT ALTER (X INT CONSTRAINT X_UNIQ UNIQUE);
```

You cannot create an index on the table ROWCTS_LIMIT since it was created with the RECORD COMMIT TIMESTAMP clause.

The following statements cannot be executed if the column table ROWCTS_CS (created with RECORD COMMIT TIMESTAMP) has a primary key or unique index:

```sql
ALTER TABLE ROWCTS_CS ROW;
CREATE TABLE ROWCTS_RS LIKE ROWCTS_CS;
```

comment_string
Specifies a descriptive comment for the table.

<comment_string> ::= <string_literal>

Specifying COMMENT <comment_string> saves you from having to execute a separate COMMENT ON statement later.

numa_node_preference_clause

Sets the NUMA node preferences. Although this clause is defined here at the table level, <numa_node_preference_clause> can be set in various locations such as range partition definitions (not hash or round-robin) and column definitions, as indicated in the syntax within the topic. <numa_node_preference_clause> is not supported for heterogeneous partitions.

<numa_node_preference_clause> ::= NUMA NODE { ( <numa_node_index_spec> )  <numa_node_index_spec> ::= <numa_node_spec> ... <range_node_spec> }  <single_node_spec> ::=  <integer_const>  <range_node_spec> ::=  <integer_const> TO <integer_const>

integer_const

<integer_const> cannot be a negative number.

numa_node_spec

Specify one or more single NUMA nodes (<single_node_spec>), or one or more NUMA node ranges (<range_node_spec>), or a mixture of both.

NUMA node indexes should be specified in the range of 0 to one less than max_numa_node_count, where max_numa_node_count is the number of NUMA nodes configured for the system. If the NUMA node index specified is greater than or equal to max_numa_node_count, then a random NUMA node in the range of 0 to one less than max_numa_node_count is selected for allocation. For example, on a system where max_numa_node_count is set to 8, if you specify a NUMA node index of 10, then any node in the range of 0 to 7 (inclusive) is chosen randomly for allocation.

Description

Tables are created without data except when <as_subquery> or <like_table_clause> is used with the WITH DATA option.

This statement is not for use with virtual tables. To create a virtual table, use the CREATE VIRTUAL TABLE statement.

Examples

Create Table A that has INTEGER-type for columns A and B. Column A has a primary key constraint.

```
CREATE ROW TABLE A (A INT PRIMARY KEY, B INT);
```
Create table T4 with group type and name:

```
CREATE ROW TABLE T4 (C1 INT PRIMARY KEY, C2 DATE) GROUP TYPE group_type1 GROUP NAME group_name1;
```

Create a history table and a column table with system versioning enabled:

```
CREATE COLUMN TABLE account_history (
    account_id INT,
    account_owner_id NVARCHAR(10),
    account_balance DOUBLE,
    valid_from timestamp NOT NULL,
    valid_to timestamp NOT NULL
);
CREATE COLUMN TABLE account (
    account_id INT PRIMARY KEY,
    account_owner_id NVARCHAR(10),
    account_balance DOUBLE,
    valid_from TIMESTAMP NOT NULL GENERATED ALWAYS AS ROW START,
    valid_to TIMESTAMP NOT NULL GENERATED ALWAYS AS ROW END,
    PERIOD FOR SYSTEM_TIME (valid_from, valid_to)
) WITH SYSTEM VERSIONING HISTORY TABLE account_history;
```

Create table T5 with identity column B that has a start value of 100 and increments by 10.

```
CREATE COLUMN TABLE T (A INT, B INT GENERATED ALWAYS AS IDENTITY (START WITH 100 INCREMENT BY 10));
```

Create a table `MyData` that has an NVARCHAR(10) column called `My_Field`, and specify a comment for both the table and the column.

```
CREATE COLUMN TABLE MyData ( My_Field NVARCHAR(10) COMMENT 'My data values')
COMMENT 'Table for data';
```

Create a table by querying another table and including an ID column:

```
CREATE TABLE BASE_TABLE (a int, b nvarchar(10));
CREATE LOCAL TEMPORARY TABLE #TARGET_TABLE (RENAME_A, RENAME_B) ADD (ID INT GENERATED ALWAYS AS IDENTITY(START WITH 1 INCREMENT BY 1)) AS (SELECT * FROM BASE_TABLE);
```

The CREATE TABLE statements in the example below create tables with names that start with "ct" and demonstrate using the WITH clause to define the columns and the data to insert:

```
CREATE TABLE t1(a INT, b INT, c INT);
CREATE TABLE t2(a INT, b INT, c INT);
INSERT INTO t1 VALUES(1,12,13);
INSERT INTO t2 VALUES(2,120,130);
CREATE TABLE ct1 AS (WITH alias AS (SELECT * FROM t1) SELECT * FROM alias);
CREATE TABLE ct2 AS (WITH w1 AS (SELECT * FROM t1) SELECT * FROM w1);
CREATE TABLE ct3 AS (WITH t1 AS (SELECT * FROM t2) SELECT * FROM t1);
CREATE TABLE ct4 AS (WITH w1 AS (SELECT * FROM t2) SELECT w1.a, t2.b FROM w1,t2);
CREATE TABLE ct5 AS (WITH w1 AS (SELECT * FROM t2), w2 AS (SELECT * FROM t1 INNER JOIN t2 ON t1.a=t2.a) SELECT w1.a FROM w1,w2);
CREATE TABLE ct6 AS (WITH d AS (SELECT 1 AS val FROM dummy) SELECT val FROM d);
```
Creates a table, t1, with annotations:

```sql
CREATE COLUMN TABLE t1(c1 INT) WITH ANNOTATIONS(
    SET 't_k1' = 't_v1', 't_k2' = 't_v2'
    COLUMN c1 SET 'c_k1' = 'c_v1', 'c_k2' = 'c_v2');
```

Creates table A1 with index index1 and then creates table A2 with index index2 like table A1 with index index1.

```sql
CREATE COLUMN TABLE a1(cola INT);
CREATE INDEX index1 ON A1 (cola);
CREATE COLUMN TABLE A2 LIKE A1 WITH DATA WITH INDEX index1 AS INDEX2;
```

**Replication**

Create table t1 with an asynchronous replica at indexserver seltera17:30040.

```sql
CREATE COLUMN TABLE t1(a INT) ASYNCHRONOUS REPLICA AT 'seltera17:30040';
```

Create table C1 that has the same definition as table A. Table C1 also has the same records as table A.

```sql
CREATE COLUMN TABLE C1 LIKE A WITH DATA;
```

Create table T4 with replicas in all indexservers.

```sql
CREATE ROW TABLE T4 (C1 INT PRIMARY KEY) REPLICA AT ALL LOCATIONS;
```

Create an asynchronous table replica REP_SCHEMA.TAB1 from the existing table SRC_SCHEMA.TAB1.

```sql
CREATE ROW TABLE REP_SCHEMA.TAB1 LIKE SRC_SCHEMA.TAB1 ASYNCHRONOUS REPLICA;
```

Create a table replica REP_TABLE1 with an asymmetric partitioning scheme from source table SRC_TABLE. All source columns are replicated. The partition scheme is HASH 8 and the partition key is C2.

```sql
CREATE COLUMN TABLE SRC_TABLE (C1 INT, C2 INT, C3 INT, primary key (C1, C2, C3)) PARTITION BY HASH (C1) PARTITIONS 32;
CREATE COLUMN TABLE REP_TABLE1 LIKE SRC_TABLE REPLICA PARTITION BY HASH (C2) PARTITIONS 8;
```

Create a replica table REP_TABLE2 with an asymmetric partitioning scheme from the source table SRC_TABLE. Only columns C1 and C3 are replicated. The partitioning scheme is RANGE and the partition key is C3.

```sql
CREATE COLUMN TABLE REP_TABLE2 LIKE SRC_TABLE REPLICA (C1, C3) PARTITION BY RANGE (C3) (PARTITION '1' <= VALUES < '100', PARTITION OTHERS);
```

The following statements create a source table and a column-wise replica. You can specify a name for the replica that you can reference later to query, drop, or move the replica.

```sql
/* create a source table, and then create a column-wise replica based on the definition of the source table */
CREATE COLUMN TABLE srcSchema.srcTableName (A INT, B INT, C NVARCHAR(10), D CHAR(1), E BIGINT, PRIMARY KEY (A));
CREATE COLUMN TABLE repSchema.repName LIKE srcSchema.srcTableName SYNCHRONOUS REPLICA (A, C, D);
```
Create table H1 that has the same definition as table H. Table H1 also has the same records as table H.
You create a column table H2 from the definition of table H without the history property and without any records.

```
CREATE HISTORY COLUMN TABLE H (A INT);
CREATE ROW TABLE H1 LIKE H WITH DATA;
CREATE ROW TABLE H2 LIKE H WITH NO DATA WITHOUT HISTORY;
```

Non-heterogeneous Partitioning

Create a non-heterogeneous range partitioned table named P1 that has a -type column A. Column U has a primary key constraint and is used as a range-partitioning column.

```
CREATE COLUMN TABLE P1 (A  PRIMARY KEY)     PARTITION BY RANGE (A)        (PARTITION '2010-02-03' <= VALUES < '2011-01-01', PARTITION VALUE = '2011-05-01');
```

Create a non-heterogeneous hash partitioned table named P2 that has INTEGER-type column I, J and K. Columns I and J make the primary key constraint and are used as hash-partitioning columns.

```
CREATE COLUMN TABLE P2 (I INT, J INT, K INT, PRIMARY KEY(I, J))      PARTITION BY HASH (I, J) PARTITIONS 2;
```

Create a non-heterogeneous hash-hash partitioned table named P3 that has INTEGER-type column I, J and K. Columns I and J make the primary key constraint and are used as hash-partitioning columns. Column K is used as a subpartitioning column.

```
CREATE COLUMN TABLE P3 (I INT, J INT, K INT, PRIMARY KEY(I, J))      PARTITION BY HASH (I, J) PARTITIONS 2, HASH (K) PARTITIONS 2;
```

Create a non-heterogeneous range-range partitioned table named P4. The OTHERS partition is defined on the first level PERSISTENT MEMORY is enabled for all partitions. The NUMA node preference is set on the first partition only.

```
CREATE COLUMN TABLE P4 (A INT, B INT)     PARTITION BY RANGE (A)
(PARTITION VALUES = 10, PERSISTENT MEMORY ON NUMA NODE ('3'),
PARTITION OTHERS PERSISTENT MEMORY ON),
(PARTITION VALUES = 20, PERSISTENT MEMORY ON;
```

Create a non-heterogeneous range-range partitioned table named P5 with dynamic range partitioning enabled.

```
CREATE COLUMN TABLE P5 (A INT, B INT NOT NULL) PARTITION BY RANGE (A)
(PARTITION VALUES = 10),
(PARTITION VALUES = 20, PARTITION OTHERS DYNAMIC THRESHOLD 2);
```

Create a non-heterogeneous range partitioned table named P6 using time selection partitioning.

```
CREATE COLUMN TABLE P6 (A INT, B INT PRIMARY KEY, _DATAAGING NVARCHAR(8))     PARTITION BY RANGE (_DATAAGING)
(USING DEFAULT STORAGE (PARTITION value = '00000000' IS CURRENT,
PARTITION '20100101' <= VALUES < '20110101', PARTITION OTHERS))
WITH PARTITIONING ON ANY COLUMNS ON
FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS OFF
FOR DEFAULT STORAGE NON CURRENT PARTITIONS PAGE LOADABLE;
```

Heterogeneous Partitioning
Create a single level range partitioned heterogeneous table. This table has no subpartitions defined.

```
CREATE COLUMN TABLE A1 (I INT, J INT, K INT, PRIMARY KEY(I, J))  
PARTITION BY RANGE (I) ((PARTITION 10 <= VALUES < 20, PARTITION 20 <= VALUES < 30) );
```

Create a multi-level range-range partitioned heterogeneous table.

```
CREATE COLUMN TABLE A2 (I INT, J INT, K INT, PRIMARY KEY(I, J))  
PARTITION BY RANGE (I) ((PARTITION 10 <= VALUES < 20, PARTITION 20 <= VALUES < 30)  
SUBPARTITION BY RANGE (J) (PARTITION 100 <= VALUES < 150, PARTITION VALUE = 200) );
```

Create a heterogeneous multi-level partitioned table. The table has three first-level partitions. The first partition has the ranges 10-20 and 20-30. The second partition has the target value 40. The final partition is OTHERS for any values outside the scope of the first to partitions. Specifies if UPDATE statements are allowed on primary key columns. The first two partitions each have a subpartition defined. Each range or target value of the first-level partitions has its own properties defined. The two subpartitions are same type (range) and reference the same column (B). The subpartitions also have dynamic range partitioning enabled.

```
CREATE COLUMN TABLE A3 (A INT, B INT NOT NULL, C INT) PARTITION BY RANGE (A)  
((PARTITION 10 <= VALUES < 20 GROUP NAME 'GRP 1' INSERT OFF, PARTITION 20 <= VALUES < 30 GROUP NAME 'GRP 1' INSERT OFF)  
SUBPARTITION BY RANGE(B) (PARTITION 0 <= VALUES < 10 PAGE LOADABLE  
GROUP SUBTYPE 'GRP 1a', PARTITION 10 <= VALUES < 100 GROUP SUBTYPE 'GRP 1b' INSERT OFF PAGE LOADABLE, PARTITION  
OTHERS DYNAMIC THRESHOLD 2),  
(PARTITION VALUES = 40 GROUP NAME 'GRP 2' INSERT OFF COLUMN LOADABLE  
SUBPARTITION BY RANGE(B) (PARTITION 0 <= VALUES < 10 GROUP SUBTYPE 'GRP 2a', PARTITION OTHERS DYNAMIC THRESHOLD 2));
```

Create a heterogeneous partitioned table with dynamic range partitioning enabled. This table has two first-level ranges. The first partition has a range of 10-20. The second partition has a target value of 30. Each partition has it own subpartition. The first subpartition has a range of 100-150. The second subpartition has a target value of 200. Both subpartitions have dynamic range partitioning enabled with a threshold of 3.

```
CREATE COLUMN TABLE T4 (A INT, B INT NOT NULL) PARTITION BY RANGE (A)  
((PARTITION 10 <= VALUES < 20) SUBPARTITION BY RANGE (B) (PARTITION 100 <= VALUES < 150, PARTITION OTHERS DYNAMIC THRESHOLD 3 ),  
(PARTITION VALUES = 30) SUBPARTITION BY RANGE (B) (PARTITION VALUES = 200, PARTITION OTHERS DYNAMIC THRESHOLD 3));
```

Create a heterogeneous partitioned table. There are three first-level (10-20, 30, and 40-50), each with different properties defined. The first two partitions share the same subpartition. There are two second-level hash partitions, each referencing the same group of columns (A, C). On the first subpartition, column A has a DATE data type, with the precision YEAR applied at the subpartition level.

```
CREATE COLUMN TABLE A5 (A DATE, B INT, C INT) PARTITION BY RANGE (B)  
((PARTITION 10 <= VALUES < 20 INSERT ON PAGE LOADABLE, PARTITION VALUES = 30 COLUMN LOADABLE GROUP NAME 'HR')  
SUBPARTITION BY HASH (YEAR(a), c) PARTITIONS 4,  
(PARTITION 40 <= VALUES < 50 PAGE LOADABLE GROUP NAME 'HR')  
SUBPARTITION BY HASH (YEAR(a), c) PARTITIONS 4);
Create a heterogeneous table with a dynamic interval on the OTHERS partition of 2 years.

```
CREATE COLUMN TABLE A3 (A INT, B DATE NOT NULL, C INT) PARTITION BY RANGE (A)    ((PARTITION 0<= VALUES < 100, PARTITION OTHERS)    SUBPARTITION BY RANGE(YEAR(B)) (PARTITION '2020' <= VALUES < '2021',     PARTITION OTHERS DYNAMIC INTERVAL 2 YEAR));
```

Data masking

Create a table that masks the data in the SSN column:

```
CREATE ROW TABLE PERSONAL_INFO (SSN VARCHAR(20)) WITH MASK (SSN USING '****');
```

Create a table that masks the data in the SSN column by session user:

```
CREATE TABLE PERSONAL_INFO (SSN VARCHAR(20)) WITH SESSION USER MASK ( SSN USING '****' );
```

Constraints

Create table C2 that has the same column data types and NOT NULL constraints as table A without any data.

```
CREATE ROW TABLE C2 AS (SELECT * FROM A) WITH NO DATA;
```

Create table F that has a foreign key that references column A of table R with update cascade option.

```
CREATE ROW TABLE R (A INT PRIMARY KEY, B NVARCHAR(10));   CREATE ROW TABLE F (FK INT, B NVARCHAR(10), FOREIGN KEY(FK) REFERENCES R ON     UPDATE CASCADE);
```

Create a self-referencing foreign key.

```
CREATE ROW TABLE SELF(A INTEGER PRIMARY KEY, B INTEGER);  ALTER TABLE SELF ADD CONSTRAINT FK_T1 FOREIGN KEY(B) REFERENCES SELF(A);
```

Create a cyclic foreign key.

```
CREATE ROW TABLE GRAND_PARENT (A INT, I INT, B INT, PRIMARY KEY (A) ); CREATE ROW TABLE PARENT (A INT, I INT, B INT, PRIMARY KEY (A) );  ALTER TABLE PARENT ADD CONSTRAINT FK_CONSTRAINT_FROM_PARENT_TO_GRAND_PARENT     FOREIGN KEY (B) REFERENCES GRAND_PARENT (A) ON DELETE CASCADE; ALTER TABLE GRAND_PARENT ADD CONSTRAINT FK_CONSTRAINT_FROM_GRAND_PARENT_TO_PARENT FOREIGN KEY (B) REFERENCES PARENT (A) ON DELETE CASCADE;
```

Schema flexibility

Create a flexible table, T.

```
CREATE COLUMN TABLE T(A INT) WITH SCHEMA FLEXIBILITY;
```

Fuzzy searching

Create table T2 with a fuzzy search index and a fuzzy search mode.

```
CREATE COLUMN TABLE T2 (KEY INT, COL1 VARCHAR(10) FUZZY SEARCH INDEX ON, COL2     NVARCHAR(10) FUZZY SEARCH MODE 'POSTCODE');
```

Temporary tables
Create a session-level global temporary table.

```
CREATE GLOBAL TEMPORARY TABLE T (C1 INT, C2 INT) ON COMMIT PRESERVE ROWS;
```

Create a transaction-level global temporary table.

```
CREATE GLOBAL TEMPORARY TABLE T (C1 INT, C2 INT) ON COMMIT DELETE ROWS;
```

Create a session-level global temporary table.

```
CREATE GLOBAL TEMPORARY TABLE T (C1 INT, C2 INT) ON COMMIT PRESERVE ROWS;
```

Create a transaction-level global temporary table.

```
CREATE GLOBAL TEMPORARY TABLE T (C1 INT, C2 INT) ON COMMIT DELETE ROWS;
```

Series

Create a non-equidistant series that records stock market trade price and volume for one year.

```
CREATE COLUMN TABLE ExampleStockTrades(
  ticker_symbol VARCHAR(5),
  trade_time TIMESTAMP,
  price DECIMAL(10,2),
  volume INTEGER
) SERIES(SERIES KEY(ticker_symbol) NOT EQUIDISTANT
  MINVALUE '2013-01-01'
  MAXVALUE '2014-01-01'
  PERIOD FOR SERIES (trade_time));
```

Create an equidistant series that covers one day intervals over a ten year period. The example table stores maximum and minimum temperatures for each day for a number of cities (identified by city_id).

```
CREATE COLUMN TABLE ExampleCityTemperature(
  city_id INTEGER,
  day_start TIMESTAMP,
  day_end TIMESTAMP,
  min_temperature DECIMAL(5,2),
  max_temperature DECIMAL(5,2)
) SERIES(SERIES KEY(city_id) EQUIDISTANT INCREMENT BY INTERVAL 1 DAY
  MINVALUE '2010-01-01'
  MAXVALUE '2020-01-01'
  PERIOD FOR SERIES (day_start, day_end));
```

Arrays

Create a new column table with an ARRAY column.

```
CREATE COLUMN TABLE T1 ( ID INT PRIMARY KEY, C1 INT ARRAY );
```

```
CREATE COLUMN TABLE T2 ( ID INT PRIMARY KEY, C1 INT ARRAY (1, 10) );
```

Create a new column table with an ARRAY column that restricts the maximum number(10) of elements.

```
CREATE COLUMN TABLE T3 ( ID INT PRIMARY KEY, C1 INT ARRAY (10) );
```

Create a new column table with an ARRAY column that restricts the minimum number(1) of elements.

```
CREATE COLUMN TABLE T4 ( ID INT PRIMARY KEY, C1 INT ARRAY (1, * ) );
```
Create a new column table with an ARRAY column that doesn’t allow duplicates of non-NULL values.

```sql
CREATE COLUMN TABLE T5 ( ID INT PRIMARY KEY, C1 INT ARRAY (1, 5) WITHOUT DUPLICATES );
```

Create a new column table with an ARRAY column that has a default value.

```sql
CREATE COLUMN TABLE T6 ( ID INT PRIMARY KEY, C1 INT ARRAY DEFAULT ARRAY() );
```

**Client-side encryption**

Create a table, T7, with a column, Name, that is encrypted using the column encryption key myCEK.

```sql
CREATE ROW TABLE T7 ( ID INT PRIMARY KEY, Name NVARCHAR(32) CLIENTSIDE ENCRYPTION ON WITH myCEK RANDOM );
```

**Persistent memory**

Create a table with PERSISTENT MEMORY enabled at the table level:

```sql
CREATE COLUMN TABLE myTable1 ( C1 INT, C2 VARCHAR (10) ) PERSISTENT MEMORY ON;
```

Create a table with PERSISTENT MEMORY enabled for the specified range partitions:

```sql
CREATE COLUMN TABLE myTable2 ( C1 INT ) PARTITION BY RANGE ( C1 ) ( PARTITION '0' <= VALUES < '10' PERSISTENT MEMORY ON, PARTITION OTHERS PERSISTENT MEMORY OFF );
```

Create a table with PERSISTENT MEMORY enabled for the specified columns:

```sql
CREATE COLUMN TABLE PMTABLE ( C1 INT PERSISTENT MEMORY ON, C2 VARCHAR (10), C3 INT PERSISTENT MEMORY OFF );
```

**NUMA node preferences examples**

Creates a column table with its NUMA-node preference set at the table level on NUMA node indexes 1, 3-5. Table T1 and all its data are allocated with a preferred NUMA allocation type on NUMA node index 1. If node 1 has an insufficient amount of memory, then SAP HANA uses the node index from the range 3-5 with the shortest distance to node 1. The preferred list of NUMA node indexes is updated in the catalog when the table is created:

```sql
CREATE COLUMN TABLE T1( colint INT, colvarchar VARCHAR(10) ) NUMA NODE ('1', '3' TO '5') ;
```

Creates a column table with its NUMA-node preference set for a partition on node index 2, and with the partition data allocated on NUMA node index 2. Data that does not belong to this partition is allocated on the node determined by the current HASH scheme available in the attribute engine:

```sql
CREATE COLUMN TABLE T1( colint INT, colvarchar VARCHAR(10) ) PARTITION BY RANGE ( colint ) ( PARTITION '0' <= VALUES < '20' NUMA NODE ('2') );
```

Creates a column table with its NUMA-node preference set for an integer column on node index 3, and with the data for this column allocated to node 3. Columns that do not have their NUMA node preference set
Inherit their preference from the table- or partition-level settings, if any partitions are set on that column. Otherwise, data is allocated with the current HASH scheme that is available in the attribute engine. The column colvarchar inherits NUMA node preference from the table-level settings (in this example, node index 5):

```sql
CREATE COLUMN TABLE T1(colint INT NUMA NODE(3), colvarchar VARCHAR(10)) NUMA NODE ('5');
```

Creates table T1 with a configuration that is similar to existing table T2, which has a NUMA NODE preference for node index 3:

```sql
CREATE COLUMN TABLE T1 LIKE T2;
```

Creates table T1 with a configuration that is similar to existing table T2 but does not include a NUMA node, so NUMA NODE preferences are not be created on T1:

```sql
CREATE COLUMN TABLE T1 LIKE T2 WITHOUT NUMA NODE;
```

**LOAD UNIT examples**

Create a table, T, that is page loadable:

```sql
CREATE COLUMN TABLE T (C1 INT, C2 VARCHAR (10)) PAGE LOADABLE;
```

**WITH ASSOCIATION**

These examples demonstrate the creation of a table with associations to other tables.

Execute the following statements to create and populate a table called t1:

```sql
CREATE TABLE t1(x1 INT, y1 INT);
INSERT INTO t1 VALUES(0, 0);
INSERT INTO t1 VALUES(1, 1);
INSERT INTO t1 VALUES(2, 2);
INSERT INTO t1 VALUES(3, 3);
```

Next, create and populate table t2 and set up an association from t2 to t1 and use the WITH DEFAULT FILTER clause to set the predicate filter to y1<3:

```sql
CREATE TABLE t2(x2 INT, y2 INT) WITH ASSOCIATIONS (JOIN t1 AS a ON a.x1 = x2 WITH DEFAULT FILTER y1 < 3);
INSERT INTO t2 VALUES(2, 0);
INSERT INTO t2 VALUES(3, -1);
INSERT INTO t2 VALUES(4, -2);
INSERT INTO t2 VALUES(5, -3);
```

Now, the following two statements return the same result (2, 2), as we have set the default predicate filter to y1<3:

```sql
SELECT * FROM t2;
SELECT * FROM t2:a[y1<3];
```

However, if you run the following statement, the result is (3, 3), as the specified filter overrides the default:

```sql
SELECT * FROM t2:a[y1>1];
```
Similarly, the following two statements return the same result, (2; None; None; None):

```
SELECT a.x1 FROM t2;
SELECT a[y1<3].x1 FROM t2;
```

However, the following statement overrides the default filter and returns (2; 3; None; None):

```
SELECT a[y1>1].x1 FROM t2;
```

Execute the following statements to create and populate table t1:

```
CREATE TABLE t1(x1 INT, y1 INT);
INSERT INTO t1 VALUES(0, 0);
INSERT INTO t1 VALUES(1, 1);
INSERT INTO t1 VALUES(2, 2);
INSERT INTO t1 VALUES(3, 3);
```

Execute the following statements to create and populate table t2 with associations to table t1 and with the default filter set to filter column values to values less than 3:

```
CREATE TABLE t2(x2 INT, y2 INT) WITH ASSOCIATIONS (JOIN t1 AS a ON a.x1 = x2
WITH DEFAULT FILTER y1 < 3);
INSERT INTO t2 VALUES(2, 0);
INSERT INTO t2 VALUES(3, -1);
INSERT INTO t2 VALUES(4, -2);
INSERT INTO t2 VALUES(5, -3);
```

Now, executing `SELECT * FROM t2:a;` returns the following:

```
2, 2
```

However, executing `SELECT * FROM t2:a[y1>1];` overrides the default filter to return the following values:

```
2, 2
3, 3
```

If you execute `SELECT a.x1 FROM t2;`, the following values are returned:

```
2,
--None
--None
--None
```

Executing `SELECT a[y1>1].x1 FROM t2;` returns the following values:

```
2,
3,
--None
--None
```

**Related Information**

- Non-heterogeneous Create Partition Clauses [page 863]
- Heterogeneous Create Partition Clauses [page 869]
4.10.1.74.1 Non-heterogeneous Create Partition Clauses

Defines the various partitioning clauses available for non-heterogeneous partitions when creating a new table.

non_heterogeneous_partitions

Partitions the table using a non-heterogeneous range, range-range, hash-range, hash-hash, or round-robin partitioning schema.

```
<non_heterogeneous_partitions> ::=   { PARTITION BY <hash_partition> [, { <subrange_partition> | <hash_partition> } ]     | PARTITION BY <range_partition> [, <subrange_partition> ]      | PARTITION BY <roundrobin_partition> [, <subrange_partition> ] } [ WITH PARTITIONING ON ANY COLUMNS { ON | OFF } ] [ FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS { ON | OFF } ] [ FOR DEFAULT STORAGE { ALL | NON CURRENT } PARTITIONS { PAGE | COLUMN } LOADABLE ]
```

Partitioning by date values requires the format YYYYMMDD, YYYY-MM-DD, or YYYY/MM/DD. Date is the most granular date time value that you can use for partitioning.

You can find more information about table partitioning in the SAP HANA Administration Guide.

hash_partition
Partitions the created table by using a hash partitioning scheme.

```plaintext
<hash_partition> ::=  
  HASH ( <partition_expression> [ , <partition_expression> [ ,... ] ] )  
  PARTITIONS { <num_partitions> | GET_NUM_SERVERS() }
```

**range_partition**

Partitions the created table by using a first-level range partitioning scheme.

```plaintext
<range_partition> ::=  
  RANGE ( <partition_expression> )  
  ( <part_range> [ , <part_range> [ ,... ] ] )
```

**subrange_partition**

Partitions the created table by using a range partitioning scheme.

```plaintext
<subrange_partition> ::=  
  RANGE ( <partition_expression> )  
  ( <part_subrange> [ , <part_subrange> [ ,... ] ] )
```

**roundrobin_partition**

Partitions the created table by using a round-robin partitioning scheme.

```plaintext
<roundrobin_partition> ::=  
  ROUNDROBIN PARTITIONS { <num_partitions> | GET_NUM_SERVERS() }
```

**partition_expression**

Declares the specifier that segregates data into partitions.

```plaintext
<partition_expression> ::=  
  <column_name> [ AS <dynamic_range_data_type> ]  
  | YEAR( <part_column_name> )  
  | MONTH( <part_column_name> )  
  | HOUR( <part_column_name> )
```

When using dynamic range partitioning on the OTHER partition, only `<column_name>` is supported.

**column_name**

Specifies the partitioning column. The column must be of a data type supported for the partitioning scheme.

**Hash partitioning**

TINYINT, SMALLINT, INT, BIGINT, DECIMAL, DECIMAL(p,s), CLOB, NCLOB, SHORTTEXT, VARCHAR, NVARCHAR, BLOB, VARBINARY, TIME, TIMESTAMP and SECONDDATE. Memory LOBs (ST_MEMORY_LOB type only) are supported but not disk LOBs.

**Range partitioning**

STRING, TINYINT, SMALLINT, INT, BIGINT (not supported for dynamic range partitioning), SHORTTEXT, NVARCHAR, DECIMAL(p,s), DATE, TIMESTAMP, SECONDDATE, FIXED, RAW (SQL Binary/Varbinary). Memory LOBs (ST_MEMORY_LOB type) are supported but not disk LOBs.

**dynamic_range_data_type**

Specifies the data type for dynamic range partitioning.

```plaintext
@dynamic_range_data_type> ::=  
  { INT | DATE }
```
YEAR / MONTH / HOUR
Specifies the precision of the date based partitioning column. When partitioning by HOUR, the <part_range> must be expressed using one of the following formats:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘YYYY-MM-DD HH’</td>
<td>• Enclosed in single quotes.</td>
</tr>
<tr>
<td></td>
<td>• Date delimiter is present.</td>
</tr>
<tr>
<td></td>
<td>• A space exists before the hour granuality.</td>
</tr>
<tr>
<td>2010-01-01 23</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘YYYYMMDDHH’</td>
<td>• Enclosed in single quotes.</td>
</tr>
<tr>
<td></td>
<td>• No date delimiter is present.</td>
</tr>
<tr>
<td></td>
<td>• No space exists before the hour granuality.</td>
</tr>
<tr>
<td>2010010123</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYYYMMDDHH</td>
<td>• Without single quotes.</td>
</tr>
<tr>
<td></td>
<td>• No date delimiter is present.</td>
</tr>
<tr>
<td></td>
<td>• No space exists before the hour granuality.</td>
</tr>
<tr>
<td>2010010123</td>
<td></td>
</tr>
</tbody>
</table>

num_partitions
Specifies the number of HASH partitions to create for the table.

<num_partitions> ::= <unsigned_integer>

GET_NUM_SERVERS()
Returns the number of servers/partitions according to table placement. You can find more information on table placement in the SAP HANA Administration Guide.

part_location
Specifies the location of each HASH partition.

<part_location> ::= { AT LOCATION ( '<HANA_host>:<HANA_port>' ) }

If not specified, locations are assigned using round robin fashion.

part_range
Specifies the range specifications for a first-level range partition.

<part_range> ::= PARTITION <range_values> [ <range_prop_list> ] [, PARTITION <range_values> [ <range_prop_list> ] [,,...] ]

part_subrange
Specifies the range specifications for a second-level range partition.

<part_subrange> ::= [ PARTITION <time_selection>, ] PARTITION <range_values> [ <range_prop_list> ] [, PARTITION <range_values> [ <range_prop_list> ] [,,...] ]

range_values
Specifies the values of the partition.

<range_values> ::=  
  <min_value> <= VALUES < <max_value>  
  | VALUE[S] = <target_value>  
  | <partition_others>

**min_value**

Specifies the minimum value of the range. The value cannot be negative.

<min_value> ::= <string_literal> | <numeric_literal>

**max_value**

Specifies the maximum value of the range. The value cannot be negative.

<max_value> ::= <string_literal> | <numeric_literal>

**target_value**

Specifies a single value of the range. The value cannot be negative.

<target_value> ::= <string_literal> | <numeric_literal>

**partition_others**

Specifies a partition for all values outside of the defined partition ranges.

<partition_others> ::=  
  OTHERS [ DYNAMIC [ THRESHOLD <threshold_count> ] ]

OTHERS can be defined on a range partition at either level.

**DYNAMIC THRESHOLD threshold_count** The keyword DYNAMIC enables dynamic range partitioning on the table (default is inactive) and THRESHOLD specifies the maximum row count in the partition before generating a new dynamic range partition from OTHERS or the priority for determining the maximum row count value. The THRESHOLD default, if not specified, is 100 000 000 rows. You can define DYNAMIC on a single RANGE partition or on the second level range of an X-RANGE partition. To enable dynamic range partitioning, the range column of OTHERS requires a NOT NULL constraint. For more information on dynamic range partitioning, see *Dynamic Range Partitioning*.

**time_selection**

Specifies the CURRENT partition for aging.

<time_selection> ::= VALUE = '00000000' IS CURRENT

CURRENT is required for tables with time selection partitioning. Time selection is supported for a single level range partitioned table or a range-range or hash-range partitioned table. Time selection is not supported for hash-hash or roundrobin-range partitioned tables. The <data_type> of the time selection column must be NVARCHAR(8). The <time_selection> clause:

- must be defined as the first partition in the range
• can reside on the first level for a single-level range partitioned table or on the second-level for range-range or hash-range partition scheme
• be a single value partition with the value 00000000

<time_selection> partitioning also requires the WITH PARTITIONING ON ANY COLUMNS clause set to ON.

range_prop_list

Specifies the properties for the partition.

\[
\text{<range_prop_list>}::=\text{<range_prop>}\ [\text{<range_prop>}\ [,\ldots]\ ]
\]
\[
\text{<range_prop>}::=\{\text{<persistent_memory_spec_clause>}\ |\ 
\text{<numa_node_preference_clause>}\}
\]

persistent_memory_spec_clause

Enables or disables persistent memory storage preference at the table, range partition, or column level, depending on where the clause is situated in the CREATE statement. For example, when specified inside the <column_definition> clause, it enables or disables persistent memory storage for the column.

\[
\text{<persistent_memory_spec_clause>}::=\text{PERSISTENT MEMORY}<\text{pm_preference}>\]
\[
\text{<pm_preference>}::=\{\text{ON}|\text{OFF}\}
\]

numa_node_preference_clause

Sets the NUMA node preferences. Although this clause is defined here at the table level, <numa_node_preference_clause> can be set in various locations such as range partition definitions (not hash or round-robin) and column definitions, as indicated in the syntax within the topic. <numa_node_preference_clause> is not supported for heterogeneous partitions.

\[
\text{<numa_node_preference_clause>}::=\text{NUMA NODE}\{\text{<numa_node_index_spec>}\}
\]
\[
\text{<numa_node_index_spec>}::=\{\text{<numa_node_spec>}\ [,\ldots]\}
\]
\[
\text{<numa_node_spec>}::=\{\text{<single_node_spec>}\ |\ \text{<range_node_spec>}\}
\]
\[
\text{<single_node_spec>}::=\text{<integer_const>}
\]
\[
\text{<range_node_spec>}::=\text{<integer_const>TO<integer_const>}
\]

\[
\text{<integer_const>}\text{ cannot be a negative number.}
\]

numa_node_spec

Specify one or more single NUMA nodes (<single_node_spec>), or one or more NUMA node ranges (<range_node_spec>), or a mixture of both.

NUMA node indexes should be specified in the range of 0 to one less than max_numa_node_count, where max_numa_node_count is the number of NUMA nodes configured for the system. If the NUMA node index specified is greater than or equal to max_numa_node_count, then a random NUMA node in the range of 0 to one less than max_numa_node_count is selected for allocation. For example, on a system where max_numa_node_count is set to 8, if you specify a NUMA node index of 10, then any node in the range of 0 to 7 (inclusive) is chosen randomly for allocation.

WITH PARTITIONING ON ANY COLUMNS

Required for time selection. Disabled (OFF) by default.

FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS
This clause is only supported for time selection and it specifies where the uniqueness check for rows entered or altered is performed. If OFF (default), then only the current (hot) partition is checked. If ON, then both current and non-current (cold) partitions are checked.

**FOR DEFAULT STORAGE {ALL | NON CURRENT} PARTITIONS {PAGE | COLUMN} LOADABLE**

This clause is only supported for time selection and specifies the load method for current and non-current partitions.

**Description**

Defines the various non-heterogeneous partitioning clauses available when creating a new table.

**Examples**

Create a non-heterogeneous range partitioned table named P1 that has a -type column A. Column U has a primary key constraint and is used as a range-partitioning column.

```sql
CREATE COLUMN TABLE P1 (A PRIMARY KEY)   
PARTITION BY RANGE (A)         
(PARTITION '2010-02-03' <= VALUES < '2011-01-01', PARTITION VALUE = 
'2011-05-01');
```

Create a non-heterogeneous hash partitioned table named P2 that has INTEGER-type column I, J and K. Columns I and J make the primary key constraint and are used as hash-partitioning columns.

```sql
CREATE COLUMN TABLE P2 (I INT, J INT, K INT, PRIMARY KEY(I, J))   
PARTITION BY HASH (I, J) PARTITIONS 2;
```

Create a non-heterogeneous hash-hash partitioned table named P3 that has INTEGER-type column I, J and K. Columns I and J make the primary key constraint and are used as hash-partitioning columns. Column K is used as a subpartitioning column.

```sql
CREATE COLUMN TABLE P3 (I INT, J INT, K INT, PRIMARY KEY(I, J))   
PARTITION BY HASH (I, J) PARTITIONS 2, HASH (K) PARTITIONS 2;
```

Create a non-heterogeneous range-range partitioned table named P4. The OTHERS partition is defined on the first level PERSISTENT MEMORY is enabled for all partitions. The NUMA node preference is set on the first partition only.

```sql
CREATE COLUMN TABLE P4 (A INT, B INT)   
PARTITION BY RANGE (A)         
(PARTITION 10 <= VALUES < 20 PERSISTENT MEMORY ON NUMA NODE ('3'), PARTITION OTHERS PERSISTENT MEMORY ON), 
RANGE (B) (PARTITION VALUES=100 PERSISTENT MEMORY ON);
```
Create a non-heterogeneous range-range partitioned table named P5 with dynamic range partitioning enabled.

```sql
CREATE COLUMN TABLE P5 (A INT, B INT NOT NULL) PARTITION BY RANGE (A) (PARTITION VALUES = 10), RANGE (B) (PARTITION VALUES = 20, PARTITION OTHERS DYNAMIC THRESHOLD 2);
```

Create a non-heterogeneous range partitioned table named P6 using time selection partitioning.

```sql
CREATE COLUMN TABLE P6 (A INT, B INT PRIMARY KEY, _DATAAGING NVARCHAR(8)) PARTITION BY RANGE (_DATAAGING) (USING DEFAULT STORAGE (PARTITION value = '00000000' IS CURRENT, PARTITION '20100101' <= VALUES < '20110101', PARTITION OTHERS)) WITH PARTITIONING ON ANY COLUMNS ON FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS OFF FOR DEFAULT STORAGE NON CURRENT PARTITIONS PAGE LOADABLE;
```

Related Information

- **Table Partitioning**
  - CREATE TABLE Statement (Data Definition) [page 825]
  - Heterogeneous Create Partition Clauses [page 869]
  - Dynamic Range Partitioning
  - TABLE_PARTITIONS System View [page 1681]

### 4.10.1.74.2 Heterogeneous Create Partition Clauses

Defines the various partitioning clauses available for heterogeneous partitions when creating a new table.

**heterogeneous_partitions**

Partitions the new table using a heterogeneous range, range-range, or range-hash partitioning scheme. For more information on partitioning schema, see **Table Partitioning** in the SAP HANA Administration Guide.

```sql
<heterogeneous_partitions> ::= PARTITION BY RANGE <partition_expression> [ [ NO ] PRIMARY KEY CHECK ] ( <part_spec> [, <part_spec> [,...] ] ) [ PRIMARY KEY UPDATE { ON | OFF } ]<part_spec> ::= { ( <part_range> [, <part_range> [,...]] ) [ <subpart_by_clause> ] | ( <part_range> ) [ <subpart_by_clause> ] }<subpart_by_clause> ::= SUBPARTITION BY { <sub_part_range> | <sub_part_hash> }
```

All subpartitions must be of the same type and reference the same column or in the case of a hash subpartition the same group of columns. Mixed range and hash subpartitions are not supported. If the first subpartition is a range using column B, then all additional subpartition must be ranges using column B. If the first subpartition is a hash, referencing columns B and C, then all subpartitions must hash referencing the same group of columns.

**partition_expression**
Declares the specifier that segregates data into partitions.

```sql
<partition_expression> ::= 
  <column_name>
  YEAR( <column_name> )
  MONTH( <column_name> )
  HOUR( <column_name> )
```

To use dynamic range partitioning, only `<column_name>` is supported for the partitioning column containing the OTHERS partition.

- **column_name** Specifies the partitioning column. The column must be of a data type supported for the partitioning scheme.

### Hash partitions

- TINYINT, SMALLINT, INT, BIGINT, DECIMAL, DECIMAL(p,s), CLOB, NCLOB, SHORTTEXT, VARCHAR, NVARCHAR, BLOB, VARBINARY, DATE, TIME, TIMESTAMP and SECONDDATE.

### Range partitions

- STRING, TINYINT, SMALLINT, INT, BIGINT (not supported for dynamic range partitioning), SHORTTEXT, NVARCHAR, DECIMAL(p,s), DATE, TIMESTAMP, SECONDDATE, FIXED, RAW (SQL Binary/Varbinary). Memory LOBs (ST_MEMORY_LOB type) are supported but not disk LOBs.

- **YEAR / MONTH** Specifies the precision of the date based partitioning column.

- **[ NO ] PRIMARY KEY CHECK**

  The primary key check is performed on the first level of a partition. If not specified, the default behavior is to not perform the check. PRIMARY KEY CHECK is not supported on multi-store tables or in combination with the time selection feature.

### part_range

Specifies the range for the first- or second-level partition.

```sql
<part_range> ::= PARTITION <range_values> [ <range_prop_list> ]
```

### range_values

Specifies the values of the first- or second-level range partition.

```sql
<range_values> ::= 
  <min_value> <= VALUES < <max_value> 
  VALUE[S] = <target_value> 
  <partition_others>
```

- **min_value** Specifies the minimum value of a first- or second-level range partition. The value cannot be negative.

```sql
<min_value> ::= <string_literal> | <numeric_literal>
```

- **max_value** Specifies the maximum value of a first- or second-level range partition. The value cannot be negative.

```sql
<max_value> ::= <string_literal> | <numeric_literal>
```
target_value
Specifies a single value of a first- or second-level range partition. The value cannot be negative.

<target_value> ::= <string_literal> | <numeric_literal>

partition_others
Specifies a partition for all values outside of the defined partition ranges.

<partition_others> ::= OTHERS [ DYNAMIC [ THRESHOLD <threshold_count> ] ] | [ DYNAMIC INTERVAL <interval> ]

DYNAMIC THRESHOLD threshold_count The keyword DYNAMIC enables dynamic range partitioning on the table (default is inactive) and THRESHOLD specifies the maximum row count in the partition before generating a new dynamic range partition from OTHERS or the priority for determining the maximum row count value. The THRESHOLD default, if not specified, is 100 000 000 rows. You can only define DYNAMIC on a single range partition or on any range subpartition. For more information on dynamic range partitioning, see Dynamic Range Partitioning.

DYNAMIC INTERVAL
Specifies a dynamic interval for the range OTHERS partition. The interval is between the last range partition and new partition created dynamically.

interval ::= <interval_value> <interval_type> 
<interval_value> ::= <unsigned_integer>
<interval_type> ::= {YEAR | MONTH | HOUR}

Dynamic interval is only supported when the partition column type is TINYINT, SMALLINT, INT, BIGINT, DATE, SECONDDATE or LONGDATE. If no <interval_type> is specified, INT is used implicitly.

range_prop_list
Specifies the first- or second-level properties for the range partition.

<range_prop_list> ::= <part_property> [ <part_property> ...]
<part_property> ::= INSERT {OFF | ON} 
| AT [ LOCATION ] ( <location> ) 
| <load_unit> 
| <group_list> 
| <persistent_memory_spec_clause>

If both LOCATION and GROUP are specified, then LOCATION takes precedence. If only GROUP is specified, then the location is determined via table placement rules. If neither is specified, then partitions inherit the location from the parent partition.

INSERT { ON | OFF }
Specifies if INSERT statements are allowed on a partition. If defined at the first-level, it applies to all second-level partitions within the first-level partition. If defined at both the first- and second-level, any value at the second-level overrides the first-level value. If not specified, default is ON.

location
Specifies where the partitions reside.

<location> ::= '<HANA_host>:<HANA_port>'

If no location is specified, then for each first-level partition, a different indexserver is selected from those available (master and slave indexservers) in a round-robin fashion, and the second-level partitions then inherit the assigned location from the parent partition.

If a location is only specified for a parent first-level partition, then any child second-level partitions inherit the location of the parent.

If a location is specified for a child second-level partition, then the second-level partition is assigned to that location, regardless of any parent location specification.

load_unit

Specifies how to load data into memory when the table is queried. Specifying the load unit is only supported on column-store tables. <load_unit> can be set at column, table, and partition levels.

<load_unit> ::= { COLUMN | PAGE } LOADABLE

COLUMN LOADABLE

In-memory loading - the entire column is loaded into memory. COLUMN LOADABLE boosts performance at the cost of higher memory usage. This is the default behavior unless another value is inherited.

PAGE LOADABLE

In-buffer cache loading - column data is loaded by page into the buffer cache. PAGE LOADABLE reduces memory usage for specific columns by not requiring those columns to be fully memory resident.

In the case of competing or unspecified <load_unit> settings, the following logic is applied.

• If there is a load unit preference specified for a column, apply that value.
• Else if there is a load unit preference specified for the parent partition, apply that value.
• Else if there is a load unit specified for the parent table, apply that value.
• Else apply the default (COLUMN LOADABLE).

group_list

Specifies the GROUP option to apply to the specified partition.

<group_list> ::= <group> [ <group> ... ]
<group> ::= GROUP {NAME | TYPE | SUBTYPE } <identifier>

Each partition can be assigned a group name, type, and subtype. If a GROUP option already exists on a partition, then the new value overwrites the existing value.

persistent_memory_spec_clause

Enables or disables persistent memory storage preference at the table, range partition, or column level, depending on where the clause is situated in the CREATE statement. For example, when specified inside the <column_definition> clause, it enables or disables persistent memory storage for the column.

<persistent_memory_spec_clause> ::= PERSISTENT MEMORY <pm_preference>
/pm_preference> ::= { ON | OFF }
**sub_part_range**

Specifies the range subpartition.

```
<sub_part_range> ::= RANGE ( <sub_range_col_def> ) { <part_range> [, <part_range> [...]] }  
<sub_range_col_def> ::= { <exist_subpart_col_name> | <partition_expression> }
```

All subpartitions must be of the same type and reference the same column or in the case of a hash subpartition the same group of columns. Mixed range and hash subpartitions are not supported. If the first subpartition is a range using column B, then all additional subpartition must be ranges using column B. If the first subpartition is a hash, referencing columns B and C, then all subpartitions must hash referencing the same group of columns.

**sub_range_col_def**

Specifies the subpartitioning column. If this is the first subpartition in the partitioned table, then specify `<partition_expression>`. For all subsequent range subpartitions added or referenced, specify `<exist_subpart_col_name>`, the name of the existing range subpartitioning column.

**sub_part_hash**

Specifies the hash subpartition.

```
<sub_part_hash> ::= HASH ( <sub_hash_col_def> ) PARTITIONS <num_partitions> [ AT [ LOCATION [ ( <loc_list> ) ] ]  
<sub_hash_col_def> ::= { <exist_col_grp_list> | <partition_expression> [, <partition_expression> [...]] ]
```

All subpartitions must be of the same type and reference the same column or in the case of a hash subpartition the same group of columns. Mixed range and hash subpartitions are not supported. If the first subpartition is a range using column B, then all additional subpartition must be ranges using column B. If the first subpartition is a hash, referencing columns B and C, then all subpartitions must hash referencing the same group of columns. With the exception of LOCATION, all hash partitions share the same properties.

**sub_hash_col_def**

Specifies the subpartitioning column. If this is the first subpartition in the partitioned table, then specify `<partition_expression>`. For all subsequent hash subpartitions added or referenced, specify `<exist_col_grp_list>`, the name of the existing hash partitioning column or group of columns.

**loc_list**

Specifies where the partitions reside.

```
<loc_list> ::= <location> [, <location> [...]] 
<location> ::= '<HANA_host>:<HANA_port>'
```

**PRIMARY KEY UPDATE { ON | OFF }**

Specifies if UPDATE statements are allowed on primary key columns. If not specified, then the default is ON.
**Description**

Defines the various heterogeneous partitioning clauses available when creating a new table.

**Examples**

Create a single level range partitioned heterogeneous table. This table has no subpartitions defined.

```sql
CREATE COLUMN TABLE A1 (I INT, J INT, K INT, PRIMARY KEY(I, J)) PARTITION BY RANGE (I) ((PARTITION 10 <= VALUES < 20, PARTITION 20 <= VALUES < 30));
```

Create a multi-level range-range partitioned heterogeneous table.

```sql
CREATE COLUMN TABLE A2 (I INT, J INT, K INT, PRIMARY KEY(I, J)) PARTITION BY RANGE (I) ((PARTITION 10 <= VALUES < 20, PARTITION 20 <= VALUES < 30) SUBPARTITION BY RANGE (J) (PARTITION 100 <= VALUES < 150, PARTITION VALUE = 200));
```

Create a heterogeneous multi-level partitioned table. The table has three first-level partitions. The first partition has the ranges 10-20 and 20-30. The second partition has the target value 40. The final partition is OTHERS for any values outside the scope of the first to partitions. Specifies if UPDATE statements are allowed on primary key columns. The first two partitions each have a subpartition defined. Each range or target value of the first-level partitions has its own properties defined. The two subpartitions are same type (range) and reference the same column (B). The subpartitions also have dynamic range partitioning enabled.

```sql
CREATE COLUMN TABLE A3 (A INT, B INT NOT NULL, C INT) PARTITION BY RANGE (A) ((PARTITION 10 <= VALUES < 20 GROUP NAME 'GRP 1' INSERT OFF, PARTITION 20 <= VALUES < 30 GROUP NAME 'GRP 1' INSERT OFF) SUBPARTITION BY RANGE (B) (PARTITION 0 <= VALUES < 10 PAGE LOADABLE GROUP SUBTYPE 'GRP 1a', PARTITION 10 <= VALUES < 100 GROUP SUBTYPE 'GRP 1b' INSERT OFF PAGE LOADABLE, PARTITION OTHERS DYNAMIC THRESHOLD 2), (PARTITION VALUES = 40 GROUP NAME 'GRP 2' INSERT OFF COLUMN LOADABLE SUBPARTITION BY RANGE (B) (PARTITION 0 <= VALUES < 10 GROUP SUBTYPE 'GRP 2a', PARTITION OTHERS DYNAMIC THRESHOLD 2));
```

Create a heterogeneous partitioned table with dynamic range partitioning enabled. This table has two first-level ranges. The first partition has a range of 10-20. The second partition has a target value of 30. Each partition has its own subpartition. The first subpartition has a range of 100-150. The second subpartition has a target value of 200. Both subpartitions have dynamic range partitioning enabled with a threshold of 3.

```sql
CREATE COLUMN TABLE T4 (A INT, B INT NOT NULL) PARTITION BY RANGE (A) ((PARTITION 10 <= VALUES < 20) SUBPARTITION BY RANGE (B) (PARTITION 100 <= VALUES < 150, PARTITION OTHERS DYNAMIC THRESHOLD 3), (PARTITION VALUES = 30) SUBPARTITION BY RANGE (B) (PARTITION VALUES = 200, PARTITION OTHERS DYNAMIC THRESHOLD 3));
```

Create a heterogeneous partitioned table. There are three first-level (10-20, 30, and 40-50), each with different properties defined. The first two partitions share the same subpartition. There are two second-
level hash partitions, each referencing the same group of columns (A, C). On the first subpartition, column A has a DATE data type, with the precision YEAR applied at the subpartition level.

```
CREATE COLUMN TABLE A5 (A DATE, B INT, C INT) PARTITION BY RANGE (B)     ((PARTITION 10 <= VALUES < 20 INSERT ON PAGE LOADABLE, PARTITION VALUES =
30 COLUMN LOADABLE GROUP NAME 'HR')
  SUBPARTITION BY HASH (YEAR(a), c) PARTITIONS 4,
(PARTITION 40 <= VALUES < 50 PAGE LOADABLE GROUP NAME 'HR')         SUBPARTITION BY HASH (YEAR(a), c) PARTITIONS 4);
```

Create a heterogeneous table with a dynamic interval on the OTHERS partition of 2 years.

```
CREATE COLUMN TABLE A3 (A INT, B DATE NOT NULL, C INT) PARTITION BY RANGE (A)    ((PARTITION 0<= VALUES < 100, PARTITION OTHERS)
  SUBPARTITON BY RANGE(YEAR(B)) (PARTITION '2020' <= VALUES < '2021',
  PARTITION OTHERS DYNAMIC INTERVAL 2 YEAR));
```

Related Information

Table Partitioning
CREATE TABLE Statement (Data Definition) [page 825]
Non-heterogeneous Create Partition Clauses [page 863]
Dynamic Range Partitioning
SAP HANA Native Storage Extension
TABLE_PARTITIONS System View [page 1681]

4.10.1.74.3 HISTORY COLUMN Option (Time Travel)

Creates a history-enabled column for use with the time travel feature.

When creating a column in a table, specify HISTORY COLUMN to create a column that enables use of the time travel feature.

Session-level time travel

A database session can be set back to a certain point-in-time. The input parameter of the statement is either a utc-timestamp or a commitid. For example:

```
SET HISTORY SESSION TO UTCTIMESTAMP = <timestamp>
<timestamp> ::= <string_literal>
```

```
SET HISTORY SESSION TO COMMIT ID = <commit_id>
<commit_id> ::= <unsigned_integer>
```

The timestamp must be in format YYYY-MM-DD HH:MM:SS[.FF7].
The COMMIT ID variant of the statement takes a commitid as a parameter. The value of the commitid parameter must occur in COMMIT_ID column of the system table SYS.TRANSACTION_HISTORY, otherwise an exception is thrown. The COMMIT ID is useful when using user defined snapshots. A user defined snapshot can be taken by storing the commitid which is assigned to a transaction during the commit phase. The commitid can be retrieved by executing the following query directly after a transaction commit:

```sql
SELECT LAST_COMMIT_ID
FROM M_TRANSACTIONS
WHERE CONNECTION_ID = CURRENT_CONNECTION;
```

To terminate a restored session to switch back to the current session, execute an explicit COMMIT or ROLLBACK on the database connection.

**Statement-level time travel**

```sql
<subquery> AS OF UTCTIMESTAMP <timestamp>
<subquery> AS OF COMMIT ID <commit_id>
```

To relate the commitid with the commit time, the SYS.TRANSACTION_HISTORY stores additional information for each transaction that commits data for history table.

For detailed information on setting session-level time travel, see the SET HISTORY SESSION statement.

- Autocommit must be turned off when a session should be restored (otherwise an exception is thrown with an appropriate error message)
- Non-history tables in restored sessions always show their current snapshot
- Only data query statement (select) is allowed inside restored sessions.
- A history table must have a primary key
- The session type can be checked from the SESSION_TYPE column of the SYS.TABLES system table
- Update on key column is not supported
- A history table cannot be imported or exported
- TEXT and SHORTTEXT columns and fulltext indexes are not supported

**Related Information**

- Data Types [page 33]
- SET HISTORY SESSION Statement (Session Management) [page 1165]
- SELECT Statement (Data Manipulation) [page 1104]

**4.10.1.75 CREATE TRIGGER Statement (Data Definition)**

Creates a trigger on a table or view.
CREATE [ OR REPLACE ] TRIGGER <trigger_name> <trigger_action_time> <trigger_event_list> ON <subject_table_name> [ REFERENCING <transition_list> ] [ <for_each> ] [ <trigger_order_clause> ] [ ONLINE ]
BEGIN [ <trigger_decl_list> ] [ <proc_handler_list> ] <trigger_stmt_list> END

Syntax Elements

OR REPLACE
Replaces the definition of the trigger if it already exists. A CREATE OR REPLACE operation does not change the ID of the trigger, nor any privileges associated with it.

trigger_name
Specifies the name of the trigger to be created, with optional schema name.

trigger_action_time
Specifies when the trigger action should occur.

BEFORE
Specifies that the trigger is executed before the DML operation on a table.

AFTER
Specifies that the trigger is executed after the DML operation on a table.

INSTEAD OF
Specifies that the trigger is executed instead of the DML operation on a view.

A view with an INSTEAD OF trigger becomes updatable. An INSTEAD OF trigger is only allowed for SQL views, not for tables or column views.

UPDATE with FROM clause and REPLACE DML is not allowed when an INSTEAD OF trigger exists.

This option is not supported within the FOR EACH clause.

trigger_event_list
Specifies the data modification command that activates the trigger action.

```sql
<trigger_event_list> ::= 
  <trigger_event> 
  | <trigger_event_list> OR <trigger_event>
<trigger_event> ::= 
  INSERT 
  | UPDATE [ [ EXCEPT ] OF <column_name_list> ]
<column_name_list> ::= <column_name> [ , <column_name> ]...
<column_name> ::= <identifier>
```

If `<column_name_list>` is provided with the UPDATE OF clause, then the UPDATE trigger only fires if a column specified in the list is updated. If `<column_name_list>` is provided with the UPDATE EXCEPT OF clause, then the UPDATE trigger is not fired if a column specified in the list is updated. The UPDATE trigger is fired regardless of the UPDATE EXCEPT OF clause if columns that are not specified in the list are updated together.

**subject_table_name**

Specifies the subject table or view name.

```sql
<subject_table_name> ::= <identifier>
```

**REFERENCING transition_list**

When a trigger transition variable is declared, the trigger can access records that are being changed by the DML triggering the trigger. For more information, see Transition variable [page 890] and Transition table [page 890].

**transition_list**

Specifies one or more transition list entries.

```sql
<transition_list> ::= <transition> [ , <transition> ]...
```

**transition**

Specifies a transition variable or table variable.

```sql
<transition> ::= <transition_variable> | <transition_table>
```

**transition_variable**

During row-level trigger execution, `<trans_var_name>.<column_name>` represents a single record from the corresponding column that is being changed by the DML.

```sql
<transition_variable> ::= { OLD | NEW } ROW [ AS ] <trans_var_name>
<trans_var_name> ::= <identifier>
```

*<column_name>* is the target table's column name. Only a ROW trigger can use a transition variable.

**transition_table**

During statement-level trigger execution, `<trans_tab_name>` represents records being changed by the trigger DML as a table variable.

```sql
<transition_table> ::= { OLD | NEW } TABLE [ AS ] <trans_tab_name>
```
Only statement triggers can use a transition table.

OLD

Specifies that you can access the old row of the DML in the trigger. This is the row that is replaced by an update or a deleted old row. UPDATE triggers and DELETE triggers can have OLD ROW transition variables or OLD TABLE transition table variables.

NEW

Specifies that you can access the new record of the DML in the trigger. This is the row that is inserted or the new updated row. UPDATE triggers and INSERT triggers can have NEW ROW transition variables or NEW TABLE transition table variables.

for_each

Specifies whether the trigger is called in a row-wise or statement-wise fashion.

The default is ROW.

ROW

Specifies that a row-level trigger is used. This is fired once for each row affected by the triggering event.

STATEMENT

Specifies that a statement-level trigger is used. This is fired once for each triggering event. Both row-store and column-store tables support statement level triggers.

trigger_order_clause

Specifies the order in which the triggers are executed.

ONLINE

Applies an intentional exclusive lock (instead of an exclusive lock) before performing the operation. An intentional exclusive lock differs from the default lock (exclusive) because it allows concurrent DML and DDL related to the object to proceed, and will retry the operation until it can proceed.

ONLINE applies an intentional exclusive lock only if the auto commit mode for DDL statements is set to on. If the auto commit mode for DDL statement is not set to on, then an exclusive lock is still applied.

trigger_decl_list

Specifies the trigger declaration.
**trigger_decl**

Declares trigger variables or trigger conditions.

```
<trigger_decl> ::=    <trigger_var_decl>     | <trigger_condition_decl>
```

The declared variable can be used as a scalar value assignment or referenced in a trigger SQL statement.

**trigger_var_decl**

Specifies the trigger variable declaration.

```
<trigger_var_decl> ::=    <var_name> [CONSTANT] <data_type> [NOT NULL] [<trigger_default_assign>];
```

**var_name**

Specifies the identifier of the trigger variable.

```
<var_name> ::= <identifier>
```

**CONSTANT**

Specifies that you cannot change the variable during trigger execution.

**data_type**

Specifies the data type of the trigger variable.

```
<data_type> ::=    DATE
                TIME
                SECONDDATE
                TIMESTAMP
                TINYINT
                SMALLINT
                INTEGER
                BIGINT
                SMALLDECIMAL
                DECIMAL
                REAL
                DOUBLE
                VARCHAR
                NVARCHAR
                ALPHANUM
                SHORTTEXT
                VARBINARY
                BLOB
                CLOB
                NCLOB
                TEXT
                BINTEXT
```

**trigger_default_assign**

Specifies the default value of the trigger variable.

```
<trigger_default_assign> ::= { DEFAULT <expression> | <expression> }
```

**trigger_condition_decl**
Specifies the condition handler declaration.

```plaintext
<trigger_condition_decl> ::=    <condition_name> CONDITION;   | <condition_name> CONDITION FOR <sql_error_code>;
```

**condition_name**

Specifies a declared condition name that you can reference in an exception handler.

```plaintext
<condition_name> ::= <identifier>
```

**sql_error_code**

Specifies the error code for exception handling.

```plaintext
<sql_error_code> ::= SQL_ERROR_CODE <int_const>
```

**proc_handler_list**

Declares exception handlers to catch SQL exceptions.

```plaintext
<proc_handler_list> ::= <proc_handler> [ <proc_handler> [...] ]
<proc_handler> ::=    DECLARE EXIT HANDLER FOR <proc_condition_value_list> <trigger_stmt>
```

**proc_condition_value_list**

Specifies one or more condition values.

```plaintext
<proc_condition_value_list> ::=    <proc_condition_value> [...]
<proc_condition_value> ::=   SQLEXCEPTION   | SQLWARNING  | <sql_error_code>    | <condition_name>
```

You can use a specific error code number or condition name declared on a condition variable.

**trigger_stmt_list**

Specifies that the trigger body syntax is a subset of the procedure body syntax.

```plaintext
<trigger_stmt_list> ::= <trigger_stmt> [ <trigger_stmt> [...] ]
<trigger_stmt> ::=    <proc_block>
    | <proc_assign>
    | <proc_if>
    | <proc_loop>
    | <proc_while>
    | <proc_for>
    | <proc_foreach>
    | <proc_signal>
    | <proc_resignal>
    | <trigger_sql>
    | <trigger_proc_call>
    | <proc_exit>
```
For more information, see the CREATE PROCEDURE statement.

**proc_block**

Specifies sections of your trigger procedures that can be nested using BEGIN and END terminals.

```sql
<proc_block> ::= BEGIN [ <trigger_decl_list> ] [ <proc_handler_list> ] <trigger_stmt_list> END;
```

**proc_assign**

Assigns values to variables.

```sql
<proc_assign> ::= <var_name> := <expression>;
<var_name> ::= <identifier>
```

**proc_if**

Controls execution flow with conditionals.

```sql
<proc_if> ::= IF <condition> THEN <trigger_stmt_list> [ <proc_elsif_list> ] [ <proc_else> ] END IF;
<proc_elsif_list> ::= <proc_elsif> [ <proc_elsif> [ ... ] ]
<proc_elsif> ::= ELSEIF <condition> THEN <trigger_stmt_list>
<proc_else> ::= ELSE <trigger_stmt_list>
```

**condition**

Specifies the conditions where the command should be performed.

```sql
<condition> ::= <condition> OR <condition>
  | <condition> AND <condition>
  | NOT <condition>
  | ( <condition> )
  | <predicate>
```

**proc_loop**

Repeatedly executes a set of trigger statements.

```sql
<proc_loop> ::= LOOP <trigger_stmt_list> END LOOP;
```

**proc_while**

Repeatedly calls a set of trigger statements while a condition is true.

```sql
<proc_while> ::= 
```
### WHILE

```sql
WHILE <condition>
    DO <trigger_stmt_list>
END WHILE;
```

### proc_for

Iterates over a set of data.

```sql
<proc_for> ::= FOR <column_name> IN [ <reverse> ] <expression> [ <expression> [... ] ]
    DO <trigger_stmt_list>
END FOR;
```

#### column_name

Specifies the name of the column where the data iteration is to occur.

```sql
<column_name> ::= <identifier>
```

#### reverse

Specifies that the results should be iterated over in reverse order.

```sql
<reverse> ::= REVERSE
```

### proc_foreach

Iterates over all elements in a set of data.

```sql
<proc_foreach> ::= FOR <column_name> AS <column_name> [ <open_param_list> ]
    DO <trigger_stmt_list>
END FOR;
```

#### open_param_list

Specifies one or more input expressions to be iterated over.

```sql
<open_param_list> ::= ( <expr_list> )
<expr_list> ::= <expression>[, <expression> [...]]
```

### proc_signal

Explicitly raises an exception from within your trigger procedures.

```sql
<proc_signal> ::= SIGNAL <signal_value> [ <set_signal_info> ];
```

### proc_resignal

Raises an exception on the action statement in an exception handler.

```sql
<proc_resignal> ::= RESIGNAL [ <signal_value> ] [ <set_signal_info> ];
```

If an error code is not specified, then RESIGNAL throws the caught exception.

#### signal_value

Specifies to SIGNAL or RESIGNAL a signal name or an SQL error code.

```sql
<signal_value> ::= { <signal_name> | <sql_error_code> }
<signal_name> ::= <identifier>
```
set_signal_info
Delivers an error message to users when specified error is thrown during trigger execution.

trigger_sql
Specifies the trigger SQL.

For information on <insert_stmt>, see the INSERT statement.
For information on <delete_stmt>, see the DELETE statement.
For information on <update_stmt>, see the UPDATE statement.
For information on <replace_stmt> and <upsert_stmt>, see the REPLACE | UPSERT statement.

select_into_stmt

trigger_proc_call
Calls a previously defined procedure.

The called procedure can only include SQL statements that are suitable for triggers.
For information on <select_list>, <from_clause>, <where_clause>, <group_by_clause>,
<having_clause>, <set_operator>, <subquery>, <order_by_clause>, and <limit>, see the
SELECT statement.

var_name_list
Specifies a scalar variable.

You can assign a selected item value to this scalar variable.
Terminates a loop.

<proc_exit> ::= BREAK;

proc_continue
Skips current loop iteration and continues with the next value.

<proc_continue> ::= CONTINUE;

proc_open
Specifies cursor operations.

<proc_open> ::= OPEN <cursor_name> [ <open_param_list> ];
<proc_fetch> ::= FETCH <cursor_name> INTO <column_name_list>;
<proc_close> ::= CLOSE <cursor_name>;

proc_return
Returns from a trigger.

<proc_return> ::= RETURN;

Cannot return a value.

Description

A trigger is a kind of procedure that automatically executes when an event occurs on a given table or view. The body of a trigger contains a set of statements that are executed when a given operation (INSERT/UPDATE/DELETE) takes place on a given subject table or subject view.

The following notes apply to triggers:

- Referencing an associated column in the trigger is not allowed because the trigger transition table only has the subject table data in context during the operation.
- In the case of batch/bulk statements, a statement-level trigger is executed once per batch entry. So for example, if inserting a batch, then the statement-level trigger is executed for every INSERT statement entry in the batch.
- Statement-level triggers are supported for both row-store and column-store tables. You can convert a row table with a statement trigger into a column table by using ALTER TABLE.
- Modification (any INSERT/UPDATE/DELETE/REPLACE) of a subject table that a trigger is defined on is not allowed in the trigger body.
- A trigger on a partitioned table cannot access the subject table, while a trigger on a non-partitioned table can execute a SELECT statement on its subject table.
- When a subject table is accessed in a trigger body, batch updates do not always show row-wise results for performance reasons.
- The maximum trigger number per single table and per DML is 1024, which means a table can have maximum 1024 INSERT triggers, 1024 UPDATE triggers, and 1024 DELETE triggers in the same time.
- Most procedures are allowed in the body of a trigger. The procedure that is called in a trigger only can have SQL statements which are suitable for a trigger body.
- Transition variable modifications are allowed in the BEFORE trigger. Internal columns ($trexkey$, $rowid$, concat column) or generated column modifications are not allowed.
Unsupported trigger actions:
- Result-set assignments, such as selecting a result set assignment into a table type
- Dynamic SQL execution, such as building SQL statements dynamically at SQLScript runtime

Permissions

To create or replace a trigger requires one of the following:
- To create a trigger, you need one of the following:
  - The TRIGGER object-level privilege granted on the `<subject_table_name>` or on the schema of the trigger.
  - CREATE ANY and TRIGGER schema level privilege granted on the schema of the trigger.
- To replace an existing trigger, you need one of the following:
  - The TRIGGER object-level privilege granted on the `<subject_table_name>` or on the schema of the trigger.
  - CREATE ANY and ALTER schema level privilege granted on the schema of the trigger.

Regardless of trigger ownership, you require the applicable privilege on each underlying object referenced by the trigger.

Examples

Basic trigger usage

Create a table that the trigger is created for.

```sql
CREATE ROW TABLE TARGET ( A INT);
```

Create a table that the trigger accesses and modifies.

```sql
CREATE ROW TABLE SAMPLE ( A INT);
```

Create the following trigger.

```sql
CREATE TRIGGER TEST_TRIGGER  AFTER INSERT ON TARGET FOR EACH ROW
BEGIN
  DECLARE SAMPLE_COUNT INT;
  SELECT COUNT(*) INTO SAMPLE_COUNT FROM SAMPLE;
  IF :SAMPLE_COUNT = 0 THEN
    INSERT INTO SAMPLE VALUES(5);
  ELSEIF :SAMPLE_COUNT = 1 THEN
    INSERT INTO SAMPLE VALUES(6);
  END IF;
END;
```

TEST_TRIGGER is executed after any record insert execution for TARGET table.

```sql
INSERT INTO TARGET VALUES (1);
```
The SAMPLE table record count is zero at the first insert attempt and the trigger TEST_TRIGGER inserts 5 into the SAMPLE table. The SELECT statement returns 5.

On the second insertion to the TARGET table, the trigger inserts 6 into the SAMPLE table because its count is now two. The SELECT statement returns 6.

**Trigger ordering**

Create the trigger TRIGGER_TEST_1 that is executed after an insert execution on TARGET table. TRIGGER_TEST_1 is executed following the execution of TEST_TRIGGER.

```sql
CREATE TRIGGER TRIGGER_TEST_1
AFTER INSERT ON TARGET
FOR EACH ROW FOLLOWS TEST_TRIGGER
BEGIN
    DECLARE SAMPLE_COUNT INT;
    SELECT COUNT(*) INTO SAMPLE_COUNT FROM SAMPLE;
    IF :SAMPLE_COUNT = 0
    THEN
        INSERT INTO SAMPLE VALUES(5);
    ELSEIF :SAMPLE_COUNT = 1
    THEN
        INSERT INTO SAMPLE VALUES(6);
    END IF;
END;
```

**Trigger with AFTER UPDATE and a WHILE loop**

```sql
CREATE ROW TABLE TARGET ( A INT);
CREATE ROW TABLE SAMPLE ( A INT);
CREATE TRIGGER TEST_TRIGGER_WHILE_UPDATE
AFTER UPDATE ON TARGET
BEGIN
    DECLARE found INT := 1;
    DECLARE val INT := 1;
    WHILE :found <> 0 DO
        SELECT count(*) INTO found FROM sample WHERE a = :val;
        IF :found = 0 THEN
            INSERT INTO sample VALUES(:val);
        END IF;
        val := :val + 1;
    END WHILE;
END;
```

**UPDATE trigger with column list**

```sql
CREATE ROW TABLE TARGET ( A INT, B INT);
CREATE ROW TABLE SAMPLE ( A INT);
CREATE TRIGGER TEST_TRIGGER_UPDATE_COLUMN_LIST
AFTER UPDATE OF B ON TARGET
BEGIN
    INSERT INTO SAMPLE VALUES(1);
END;
```

UPDATE TARGET SET A = 0;
-- does not fire the trigger
SELECT COUNT(*) FROM SAMPLE;
The SELECT statement returns 0.

```
UPDATE TARGET SET B = 0; -- does fire the trigger
SELECT COUNT(*) FROM SAMPLE;
```

The SELECT statement returns 1.

**UPDATE EXCEPT OF trigger with column list**

```
CREATE ROW TABLE TARGET ( A INT, B INT);
CREATE ROW TABLE SAMPLE ( A INT);
CREATE TRIGGER TEST_TRIGGER_UPDATE_EXCEPT_OF_COLUMN_LIST
AFTER UPDATE EXCEPT OF B ON TARGET
BEGIN
    INSERT INTO SAMPLE VALUES(1);
END;
```

```
INSERT INTO TARGET VALUES(1, 1);
UPDATE TARGET SET B = 0; -- does not fire the trigger
SELECT COUNT(*) FROM SAMPLE;
```

The SELECT statement returns 0.

```
UPDATE TARGET SET A = 0; -- does fire the trigger
SELECT COUNT(*) FROM SAMPLE;
```

The SELECT statement returns 1.

```
UPDATE TARGET SET A = 2, B = 2; -- does fire the trigger
SELECT COUNT(*) FROM SAMPLE;
```

The SELECT statement returns 2.

**Trigger with an AFTER INSERT and FOR loop**

```
CREATE ROW TABLE TARGET ( A INT);
CREATE ROW TABLE control_tab(id INT PRIMARY KEY, name VARCHAR(30), payment INT);
CREATE ROW TABLE message_box(message VARCHAR(200), log_time TIMESTAMP);
CREATE TRIGGER TEST_TRIGGER_FOR_INSERT
AFTER INSERT ON TARGET
BEGIN
    DECLARE v_id        INT := 0;
    DECLARE v_name      VARCHAR(20) := '';
    DECLARE v_pay       INT := 0;
    DECLARE v_msg       VARCHAR(200) := '';
    DELETE FROM message_box;
    FOR v_id IN 100 .. 103 DO
        SELECT name, payment INTO v_name, v_pay FROM control_tab WHERE id = :v_id;
        v_msg := :v_name || ' has ' || TO_VARCHAR(:v_pay);
        INSERT INTO message_box VALUES (:v_msg, CURRENT_TIMESTAMP);
    END FOR;
END;
```

**Trigger with EXIT HANDLER examples**

```
CREATE ROW TABLE TARGET ( A INT);
CREATE ROW TABLE MYTAB (I INTEGER PRIMARY KEY);
CREATE TRIGGER MYTRIG_SQLEXCEPTION
AFTER INSERT ON TARGET
BEGIN
```

SAP HANA SQL Reference Guide for SAP HANA Platform

888  PUBLIC

SQL Reference
DECLARE EXIT HANDLER FOR SQL_EXCEPTION RESIGNAL;
INSERT INTO MYTAB VALUES (1);
INSERT INTO MYTAB VALUES (1);  -- expected unique violation error: 301
-- not reached
END;

CREATE TRIGGER MYTRIG_SQL_ERROR_CODE
AFTER UPDATE ON TARGET
BEGIN
DECLARE EXIT HANDLER FOR SQL_ERROR_CODE 301 RESIGNAL;
INSERT INTO MYTAB VALUES (1);
INSERT INTO MYTAB VALUES (1);  -- expected unique violation error: 301
-- not reached
END;

CREATE TRIGGER MYTRIG_CONDITION
AFTER DELETE ON TARGET
BEGIN
DECLARE MYCOND CONDITION FOR SQL_ERROR_CODE 301;
DECLARE EXIT HANDLER FOR MYCOND RESIGNAL;
INSERT INTO MYTAB VALUES (1);
INSERT INTO MYTAB VALUES (1);  -- expected unique violation error: 301
-- not reached
END;

CREATE ROW TABLE TARGET (A INT);
CREATE ROW TABLE MYTAB (I INTEGER PRIMARY KEY);
CREATE ROW TABLE MYTAB_TRIGGER_ERR (err_code INTEGER, err_msg VARCHAR(30));
CREATE TRIGGER MYTRIG_SIGNAL
AFTER INSERT ON TARGET
BEGIN
DECLARE MYCOND CONDITION FOR SQL_ERROR_CODE 10001;
DECLARE EXIT HANDLER FOR MYCOND INSERT INTO MYTAB_TRIGGER_ERR VALUES
(::SQL_ERROR_CODE, ::SQL_ERROR_MESSAGE);
INSERT INTO MYTAB VALUES (1);
SIGNAL MYCOND SET MESSAGE_TEXT = 'my error message1';  -- signal immediately
-- not reached
INSERT INTO MYTAB VALUES (2);
END;

INSERT INTO SUBJECT VALUES (1)
SELECT * FROM MYTAB_TRIGGER_ERR;
(10001, my error message1)
SELECT * FROM MYTAB;

The SELECT statement returns 1.

CREATE TRIGGER MYTRIG_RESIGNAL
AFTER UPDATE ON TARGET
BEGIN
DECLARE MYCOND CONDITION FOR SQL_ERROR_CODE 10002;
DECLARE EXIT HANDLER FOR MYCOND RESIGNAL; -- 2. throws error with error
code 10002
INSERT INTO MYTAB VALUES (1);
SIGNAL MYCOND SET MESSAGE_TEXT = 'my error message2'; -- 1. signal
immediately
-- not reached
INSERT INTO MYTAB VALUES (2);
END;
UPDATE SUBJECT SET A = 100 WHERE A = 1;
ERR-10002  ERR-MSG:user-defined error: "TRIGGER_EXCEPTION"."MYTRIG_RESIGNAL": line 4 col 37 (at pos 210): [10002] (range 3) user-defined error exception: my error message2
Row-wise trigger using transition variable

```
CREATE ROW TABLE TARGET ( A INT, B VARCHAR(10));
CREATE ROW TABLE SAMPLE_OLD ( A INT, B VARCHAR(10));
CREATE ROW TABLE SAMPLE_NEW ( A INT, B VARCHAR(10));
CREATE ROW TABLE SAMPLE ( A INT, B VARCHAR(10));
INSERT INTO TARGET VALUES ( 1, 'oldvalue');
INSERT INTO TARGET VALUES ( 2, 'oldvalue');
INSERT INTO TARGET VALUES ( 5, 'oldvalue');
CREATE TRIGGER TEST_TRIGGER_VAR_UPDATE
AFTER UPDATE ON TARGET
REFERENCING NEW ROW mynewrow, OLD ROW myoldrow
FOR EACH ROW
BEGIN
  INSERT INTO SAMPLE_new VALUES(:mynewrow.a, :mynewrow.b);
  INSERT INTO SAMPLE_old VALUES(:myoldrow.a, :myoldrow.b);
  INSERT INTO SAMPLE VALUES(0, 'trigger');
END;
UPDATE TARGET SET b = 'newvalue' WHERE A < 3;
SELECT * FROM TARGET;
1, 'newvalue'
2, 'newvalue'
5, 'oldvalue'
SELECT * FROM SAMPLE_NEW;
1, 'newvalue'
2, 'newvalue'
SELECT * FROM SAMPLE_OLD;
1, 'oldvalue'
2, 'oldvalue'
SELECT * FROM SAMPLE;
0, 'trigger'
0, 'trigger'
```

Statement-wise trigger using transition table

```
CREATE ROW TABLE TARGET ( A INT, B VARCHAR(10));
CREATE ROW TABLE SAMPLE_OLD ( A INT, B VARCHAR(10));
CREATE ROW TABLE SAMPLE_NEW ( A INT, B VARCHAR(10));
CREATE ROW TABLE SAMPLE ( A INT, B VARCHAR(10));
INSERT INTO TARGET VALUES ( 1, 'oldvalue');
INSERT INTO TARGET VALUES ( 2, 'oldvalue');
INSERT INTO TARGET VALUES ( 5, 'oldvalue');
CREATE TRIGGER TEST_TRIGGER_TAB_UPDATE
AFTER UPDATE ON TARGET
REFERENCING NEW TABLE mynewtab, OLD TABLE myoldtab
FOR EACH STATEMENT
BEGIN
  INSERT INTO SAMPLE_new SELECT * FROM :mynewtab;
  INSERT INTO SAMPLE_old SELECT * FROM :myoldtab;
  INSERT INTO SAMPLE VALUES(0, 'trigger');
END;
UPDATE TARGET SET b = 'newvalue' WHERE A < 3;
SELECT * FROM TARGET;
1, 'newvalue'
2, 'newvalue'
5, 'oldvalue'
SELECT * FROM SAMPLE_NEW;
1, 'newvalue'
2, 'newvalue'
SELECT * FROM SAMPLE_OLD;
1, 'oldvalue'
2, 'oldvalue'
SELECT * FROM SAMPLE;
0, 'trigger'
CREATE ROW TABLE T1(A INTEGER PRIMARY KEY, B INTEGER);
CREATE COLUMN TABLE T2(A INTEGER PRIMARY KEY, C INTEGER);
CREATE VIEW V1 AS (SELECT T1.A, B, C, T1.A+10 D FROM T1, T2 WHERE T1.A = T2.A);
```
CREATE TRIGGER TR1 INSTEAD OF INSERT ON V1 REFERENCING NEW ROW NEW
BEGIN
  INSERT INTO T1 VALUES(:NEW.A, :NEW.B);
  INSERT INTO T2 VALUES(:NEW.A, :NEW.C);
END;
INSERT INTO V1(A,B,C) VALUES(1,2,3);
SELECT FROM T1 ORDER BY A, B;
The SELECT statement returns 1, 2.

SELECT FROM T2 ORDER BY A, C;
The SELECT statement returns 1, 3.

SELECT FROM V1 ORDER BY A, B, C, D;
The SELECT statement returns 1, 2, 3, 11

CREATE TRIGGER TR2 INSTEAD OF UPDATE ON V1 REFERENCING OLD ROW OLD, NEW ROW NEW
BEGIN
  UPDATE T1 SET A=:NEW.A, B=:NEW.B WHERE A=:OLD.A;
  UPDATE T2 SET A=:NEW.A, C=:NEW.C WHERE A=:OLD.A;
END;
UPDATE V1 SET A = 5, B = 6, C = 7 WHERE A = 1;
SELECT FROM T1 ORDER BY A, B;
The SELECT statement returns 5, 6.

SELECT FROM T2 ORDER BY A, C;
The SELECT statement returns 5, 7.

SELECT FROM V1 ORDER BY A, B, C, D;
The SELECT statement returns 5, 6, 7, 11.

CREATE TRIGGER TR3 INSTEAD OF DELETE ON V1 REFERENCING OLD ROW OLD
BEGIN
  DELETE FROM T1 WHERE A = :OLD.A;
  DELETE FROM T2 WHERE A = :OLD.A;
END;
DELETE FROM V1;
SELECT FROM T1 ORDER BY A, B;
SELECT FROM T2 ORDER BY A, C;
SELECT FROM V1 ORDER BY A, B, C, D;

Related Information

TRIGGERS System View [page 1694]
Table Variable Type Definition
Introduction to SQL [page 27]
Data Types [page 33]
Expressions [page 56]
CREATE PROCEDURE Statement (Procedural) [page 776]
Predicates [page 61]
4.10.1.76  CREATE TYPE Statement (Procedural)

Creates a user-defined type

Syntax

CREATE TYPE <type_name>
  AS TABLE (<column_definition>[{,<column_definition>}...]) [ SQLSCRIPT SEARCH
KEY( <column_name> [,…] ) ]

Syntax Elements

**type_name**

Identifies the table type to be created and, optionally, in which schema the creation should take place.

```plaintext
<type_name> ::= [<schema_name>.]<identifier>
<schema_name> ::= <unicode_name>
```

**column_definition**

Defines a table column.

```plaintext
<column_definition> ::= <column_name> <data_type> [<column_store_data_type>]
```

**column_name**

Specifies the table column name.

```plaintext
<column_name> ::= <identifier>
```

**data_type**

Specifies the data type for the column.

```plaintext
<data_type> ::= DATE | TIME | SECONDDATE | TIMESTAMP | TINYINT
  | SMALLINT | INTEGER | BIGNIT | SMALLDECIMAL | DECIMAL
  | REAL | DOUBLE | VARCHAR | NVARCHAR | ALPHANUM
  | VARBINARY | BLOB | CLOB | NCLOB | SHORTTEXT
  | TEXT | BINTEXT
```
SQLSCRIPT SEARCH KEY clause

Specifies sorting instructions for the table. For use with SQLScript procedures. See the SAP HANA SQLScript Guide for more information.

Description

Use a user-defined type in the CREATE PROCEDURE and CREATE FUNCTION commands.

The syntax for defining table types follows the SQL syntax for defining new types. The table type is specified using a list of attribute names and primitive data types. For each table type, attributes must have unique names.

Example

Create a table type called tt_publishers.

```
CREATE TYPE tt_publishers AS TABLE (    publisher INTEGER,    name VARCHAR(50),    price DECIMAL,    cnt INTEGER);
```

Related Information

Sorted Table Variables
CREATE PROCEDURE Statement (Procedural) [page 776]
Data Types [page 33]
CREATE FUNCTION Statement (Procedural) [page 752]

4.10.1.77  CREATE USER Statement (Access Control)

Creates a new database user.
Syntax

```
CREATE [ RESTRICTED ] USER <user_name>
  [ <authentication_options> ]
  [ <validity_specification> ]
  [ <set_user_parameters> ]
  [ <ldap_group_authorization> ]
  [ <usergroup_membership_option> ]
```

Syntax Elements

**user_name**

Specifies the user name of the user to be created.

```
<user_name> ::= <unicode_name>
```

**authentication_options**

Specifies the authentication options for the user.

```
<authentication_options> ::= 
  { <remote_identity_option> 
    | <password_option> 
    | <external_ident> 
    | <with_ident_opts> 
  } 
```

**remote_identity_option**

Specifies a database user in another tenant database as the remote identity of the database user being created.

```
<remote_identity_option> ::= 
  <remote_database_identity> [ <remote_database_identity> [...] ]
  <remote_database_identity> ::= <remote_user_name> AT DATABASE <database_name>
  <remote_user_name> ::= <simple_identifier>
  <database_name> ::= <identifier>
```

**password_option**

Specifies the user password of the user being created.

```
<pASSWORD>_option> ::= PASSWORD <password> [ NO FORCE_FIRST_PASSWORD_CHANGE ]
```

The option NO FORCE_FIRST_PASSWORD_CHANGE overrules the setting of the password policy parameter force_first_password_change and allows the final password to be specified during user creation.

The password must follow the rules defined for the current SAP HANA instance. The password rules include a minimum password length and a definition of which character types (lower, upper, digit, special characters) must be part of the password.

**external_ident**
Specifies the external identity for the user. This clause is only for Kerberos authentication.

<external_ident> ::= IDENTIFIED EXTERNALLY AS <external_identity>

**external_identity**

Defines an external identity for user authentication.

<external_identity> ::= <simple_identifier> | <string_literal>

**with_ident_opts**

Defines a method for user authentication.

<with_ident_opts> ::= WITH IDENTITY <provider_identity> [ ]

<provider_identity> ::= <saml_provider_ident> | <x509_provider_ident> | <kerberos_provider_ident> | <logon_ticket_ident> | <assertion_ticket_ident> | <jwt_provider_ident> | <ldap_provider_ident>

**saml_provider_ident**

Defines a SAML provider.

<saml_provider_ident> ::= <mapped_user_name> FOR SAML PROVIDER

<saml_provider_name> ::= <simple_identifier>

<mapped_user_name> specifies the mapped SAML user name to use. If the keyword ANY is used, then the SPProvidedID attribute of the SAML assertion contains the name of the database user that the assertion is valid for. For more information, see the section on single sign-on using SAML in the SAP HANA Security Guide.

<saml_provider_name> specifies the identifier of a SAML provider that already exists in the system.

**x509_provider_ident**

Defines an X.509 certificate issuer or X.509 provider.

<x509_provider_ident> ::=<subject_distinguished_name> ISSUER <issuer_distinguished_name> FOR X509

[ ] [ ]

<subject_distinguished_name> ::= <string_literal> | <simple_identifier>

<issuer_distinguished_name> ::= <string_literal> | <simple_identifier>

<subject_distinguished_name> specifies the subject name provided in the certificate of the X.509 identity provider.

<issuer_distinguished_name> specifies the issuer name provided in the certificate of the X.509 identity provider.
You can specify a wildcard (ANY) for the subject name only for an X.509 provider with matching rules.

<\x509_provider_name> specifies the identifier of an X.509 provider that already exists in the system.

**kerberos_provider_id**

Defines a KERBEROS identity.

```
<kerberos_provider_ident> ::= <kerberos_principal_name> FOR KERBEROS
<kerberos_principal_name> ::= <string_literal>
```

<kerberos_principal_name> specifies an identity within an external authentication system.

**logon_ticket_ident**

Sets SAP Logon Ticket as the authentication method.

```
<logon_ticket_ident> ::= FOR SAP LOGON TICKET
```

**assertion_ticket_ident**

Sets SAP Assertion Ticket as the authentication method.

```
<assertion_ticket_ident> ::= FOR SAP ASSERTION TICKET
```

**jwt_provider_id**

Defines a JWT provider-user-mapping.

```
<jwt_provider_ident> ::= <mapped_user_name> FOR JWT PROVIDER
<mapped_user_name> ::= { ANY | <string_literal> }
<jwt_provider_name> ::= <simple_identifier>
```

<mapped_user_name> specifies the mapped JWT user name to use. If the keyword ANY is used, then the JWT token will contain the name of the database user that the token is valid for.

<jwt_provider_name> specifies the identifier of a JWT provider that already exists in the system.

**ldap_provider_ident**

Creates a user enabled to use LDAP authentication.

```
<ldap_provider_ident> ::= FOR LDAP PROVIDER
```

**validity_specification**

Defines the validity specification with an optional user parameter option.

```
<validity_specification> ::= VALID <validity_opts>
```

**validity_opts**

Configures the user's temporal validity.

```
<validity_opts> ::=<from_specification> [ <until_specification> ]
| <until_specification>
```
The specified user can only connect within the given date range. By using this feature, you can create users in advance or restrict the period when users can connect to the SAP HANA database.

from_specification

Sets when the user is valid from.

<from_specification> ::= FROM { <timestamp> | NOW }

Use the NOW keyword to configure the user account to be valid from the current time.

The default value is NOW.

until_specification

Sets when the user is valid until.

<until_specification> ::= UNTIL { <timestamp> | FOREVER }

Use the FOREVER keyword to configure the user account to never expire.

The default value is FOREVER.

timestamp

Specifies a timestamp.

<timestamp> ::= <string_literal>

set_user_parameters

Sets parameters in the user parameter list.

<set_user_parameters> ::= SET PARAMETER <user_parameter_list>

user_parameter_list

Specifies a list of user parameters.

<user_parameter_list> ::= <user_parameter> [, <user_parameter> [,…] ]

user_parameter

Defines user parameters.

$user_parameter$ ::= CLIENT = <string_literal> | LOCALE = <string_literal> | TIME ZONE = <string_literal> | EMAIL ADDRESS = <string_literal> | STATEMENT MEMORY LIMIT = '<integer>' | STATEMENT THREAD LIMIT = '<integer>' | STATEMENT THREAD LIMIT = '<integer>'
### User Parameter | Purpose
--- | ---
**CLIENT** | When you define column store views, the user parameter CLIENT restricts the access of the user to the specified client. This parameter is for internal use only.

**LOCALE** | When you define column store views, the user parameter LOCALE translates information according to the user’s locale.

**TIME ZONE** | Not used by the SAP HANA database, but can be read by external applications.

**EMAIL ADDRESS** | Not currently used by the SAP HANA database, but can be read by external applications. This value must be unique.

**STATEMENT MEMORY LIMIT** | Sets a user specific statement memory limit in gigabytes.
- If both a global and a user statement memory limit are set, then the user-specific limit takes precedence, regardless of whether it is higher or lower than the global statement memory limit.
- If the user-specific statement memory limit is removed, then the global limit takes effect for the user.
- Setting the statement memory limit to 0 disables any statement memory limit for the user.
- For STATEMENT MEMORY LIMIT to take effect, resource_tracking and memory_tracking must be active.

**STATEMENT THREAD LIMIT** | Sets a user-specific concurrency limit on statements (despite the name, STATEMENT THREAD LIMIT is not an actual thread limit). Similar behaviors to STATEMENT MEMORY LIMIT apply to STATEMENT THREAD LIMIT.

**PARAMETER PRIORITY** | Sets a user-level priority value for all statements in the current connection; the range of possible values is from 0 to 9 (the default is 5).

**ldap_group_authorization**

Specifies the LDAP group authorization mode for the user.

```sql
<ldap_group_authorization> ::= AUTHORIZATION { LOCAL | LDAP }

LOCAL
```
Specifies the use of local authorization mode for the user. This is the default.

When changing the LDAP authorization mode from LDAP to LOCAL for a user, roles that were granted using LDAP are revoked, and the DN value for the SAP HANA user is cleared.

**LDAP**

Specifies the use of LDAP authorization mode for the user.

Users with LDAP authorization mode cannot be granted permissions directly, and cannot have local roles granted to them.

When changing the LDAP authorization mode from LOCAL authorization to LDAP for a user, LDAP roles are evaluated and activated upon next login. Also, local roles and privileges granted to the user are revoked except for roles that are granted by SYS—for example, the PUBLIC role—and the CREATE ANY privilege on the user’s own schema.

**usergroup_membership_option**

Adds the user to the specified user group.

```
SET USERGROUP <usergroup_name>
```

You must have the USER ADMIN privilege to add a user to a user group, unless the user group is defined as DISABLE USERGROUP OPERATOR, in which case you must have the USERGROUP OPERATOR privilege on `<usergroup_name>`.

When adding a user to a user group, you must have the appropriate privileges for the user group you are adding to, as follows:

- For user groups defined as DISABLE USER ADMIN, you must have USERGROUP OPERATOR privilege on the user group to add users.
- For user groups that are not defined as DISABLE USER ADMIN, you must have USER ADMIN privilege, or USERGROUP OPERATOR privilege on the user group, to add users.

**Description**

The specified user name must not be identical to the name of an existing user, usergroup, role, or schema.

There are some users that are delivered with the SAP HANA database: SYS, SYSTEM, and a number of users with the _SYS_ prefix.

Users in the database can be authenticated by varying mechanisms:

- Internal authentication mechanism using a password.
- External mechanisms such as Kerberos, SAML, SAP Logon Ticket, SAP Assertion Ticket, or X.509.

A user can be authenticated by more than one mechanism at a time; however, only one password and one principal name for Kerberos can be valid at any one time.

Specify at least one authentication mechanism to allow the user to connect and work with the database instance. For backwards compatibility, the IDENTIFIED EXTERNALLY AS `<external_identity>` syntax is maintained. This syntax essentially performs the same function as the `<kerberos_principal_name>` FOR KERBEROS syntax.
External users are authenticated by using an external system, for example a Kerberos system. For detailed information about external identities, contact your domain administrator.

A schema with the user’s name is created for each database user: this schema cannot be explicitly dropped. The user’s schema is automatically dropped when the user is deleted. The database user owns this schema and it is used as their default schema when they execute a command without explicitly specifying a schema name.

Users created without the RESTRICTED option can create any object in their own schema and they are granted the role PUBLIC allowing them to select any system view. Users created with the RESTRICTED option have no privileges, not even the ability to create objects in their own schema. These users are meant for using specific applications only. All privileges needed for the application have to be granted to such users, preferably using a role combined with the application.

Permissions

Only database users with the USER ADMIN privilege, or USERGROUP OPERATOR privilege on a user group, are allowed to create a database user.

Configuration Parameters

Configuration parameters concerning the user’s password can be observed with the monitoring view M_PASSWORD_POLICY. These parameters are stored in the password policy section of the indexserver.ini file.

The description of the parameters concerned can be found in the Appendix of the SAP HANA Security Guide under Password Policy Parameters.

Examples

Example 1 - Create user with password

The following example shows you how to create a user T12345 with a password Password123.

```
CREATE USER T12345 PASSWORD Password123;
```
Example 2 - Create user that uses an external authentication mechanism

The following example creates an SAML provider named `ac_saml_provider` in the database, specifying a subject and issuer for `ACompany`.

```
CREATE SAML PROVIDER ac_saml_provider
    WITH SUBJECT 'CN = wiki.detroit.ACompany.corp,OU = ACNet,O = ACompany,C = EN'
    ISSUER 'E = John.Do@acompany.com,CN = ACNetCA,OU = ACNet,O = ACompany,C = EN';
```

The following example creates a new user called `new_user` with password `Password1`. The user can connect to the system by using the given password and with an assertion of the existing SAML provider `ac_saml_provider`. The `<mapped_user_name>` is set to `ANY` as the assertion provides the database user name.

```
CREATE USER new_user PASSWORD Password1 WITH IDENTITY ANY FOR SAML PROVIDER ac_saml_provider;
```

The following example creates a user that is only allowed to log in using X509:

```
CREATE USER testuser WITH IDENTITY 'C=US, ST=VA, L=Fairfax, O=example.com, CN=testuser' ISSUER 'C=US, ST=VA, L=Fairfax, O=example.com, CN=Root CA' FOR X509;
```

The following example creates a user that is only allowed to log in using password or X509:

```
CREATE USER testuser PASSWORD BadPassword5 WITH IDENTITY 'C=US, ST=VA, L=Fairfax, O=example.com, CN=testuser' ISSUER 'C=US, ST=VA, L=Fairfax, O=example.com, CN=Root CA' FOR X509;
```

The following example creates a user that is only allowed to log in using X509 and whose account is deactivated on Jan 1, 2028:

```
CREATE USER testuser WITH IDENTITY 'C=US, ST=VA, L=Fairfax, O=example.com, CN=testuser' ISSUER 'C=US, ST=VA, L=Fairfax, O=example.com, CN=Root CA' FOR X509
    VALID UNTIL '2028-01-01';
```

The following example creates a user that is only allowed to log in using SAML (this statement actually throws an error if the SAML provider does not exist):

```
CREATE USER testuser WITH IDENTITY ANY FOR SAML PROVIDER nonexistandsamlprovider;
```

Example 3 - Create user that is associated with a remote user

This example creates a new user called `USER1` associated with the user `USER2` at the remote database `DB2`.

```
CREATE USER USER1 WITH REMOTE IDENTITY USER2 AT DATABASE DB2;
```
Related Information

Unpermitted Characters in User Names
CREATE SAML PROVIDER Statement (Access Control) [page 803]
CREATE JWT PROVIDER Statement (Access Control) [page 767]
Single Sign-On Using SAML 2.0
M_PASSWORD_POLICY System View [page 2039]
USERS System View [page 1698]
USER_PARAMETERS System View [page 1702]
INVALID_CONNECT_ATTEMPTS System View [page 1588]
SAML_PROVIDERS System View [page 1648]
SAML_USER_MAPPINGS System View [page 1649]

4.10.1.78  CREATE USERGROUP Statement (Access Control)

Creates a usergroup.

Syntax

```
CREATE USERGROUP <usergroup_name>   
  [ DISABLE USER ADMIN ]  
  [ NO GRANT TO CREATOR ]  
  [ SET PARAMETER <parameter_key_value_list> ]  
  [ ENABLE PARAMETER SET <parameter_set_name> ]  
  [ DISABLE CLIENT CONNECT ]
```

Syntax Elements

`usergroup_name`

Specifies a name for the usergroup. `<usergroup_name>` cannot match the name of an existing schema, user, usergroup, or role.

`DISABLE USER ADMIN`

Prevents users with USER ADMIN privilege from administering the usergroup (for example, changing settings and adding/removing users).

If `DISABLE USER ADMIN` is specified, then only users with USERGROUP OPERATOR privilege on the usergroup can administer the usergroup.

`NO GRANT TO CREATOR`

Prevents the automatic granting of all object privileges of the usergroup to the user creating the usergroup.

`SET PARAMETER parameter_key_value_list`
Configures parameter options as a key-value list for a parameter set that can be enabled and disabled for the group. Specifically, use this clause to configure group-specific values for the individual options of the password policy set called 'password policy'.

When the SET PARAMETER clause is used to specify password policy options, all policy option settings are stored for the usergroup with the exception of password_lock_for_system_user, which is not applicable or allowed for usergroups. However, this does not mean you need to specify all of the password policy options. Password policy options that are not specified in the SET PARAMETER clause are copied from the current system-wide password policy settings and stored with the specified options for the usergroup. Subsequent changes to the system-wide settings do not impact any stored password policy options for the usergroup. Refer to the SAP HANA Administration Guide for the list of password policy options you can set.

Once you set the parameter set options, use a ALTER USERGROUP...SET PARAMETER statement to alter specific parameter settings without impacting other parameter settings that have been set.

**ENABLE PARAMETER SET parameter_set_name**

Enables the specified parameter set. The only supported parameter set name is 'password policy'.

Enabling the password policy parameter causes any password policy options that have been defined to be applied. If ENABLE PARAMETER SET 'password policy' is not specified, any password policy options configured using the SET PARAMETER clause remain configured for the usergroup, but not applied. Also, if a parameter set is enabled before any parameters have been set, all parameter values are identical to the current system wide settings.

**DISABLE CLIENT CONNECT** Prevents users in the usergroup from connecting to the database.

**Description**

If you create the usergroup with DISABLE USER ADMIN, then you must use the GRANT statement to grant USERGROUP OPERATOR privilege to a user on the usergroup so that they can administer it.

You add users to a usergroup using the CREATE USER and ALTER USER statements.

The M_EFFECTIVE_PASSWORD_POLICY system view can be used to query the current password policy settings for a user.

**Permissions**

You must have the USER ADMIN privilege to create a usergroup.
Example

The following example creates a usergroup called MyUserGroup and sets the restriction that only users with USERGROUP OPERATOR privilege on the usergroup can administer it:

```sql
CREATE USERGROUP MyUserGroup DISABLE USER ADMIN NO GRANT TO CREATOR;
```

The following example creates a usergroup called MyUserGroup and sets several password policy options for the usergroup:

```sql
CREATE USERGROUP MyUserGroup SET PARAMETER 'password_layout' = 'A1a!',
'minimal_password_length' = '16' ENABLE PARAMETER SET 'password policy';
```

The following example creates a usergroup called MyUserGroup and disables connections to the database by its members:

```sql
CREATE USERGROUP MyUserGroup DISABLE CLIENT CONNECT;
```

Related Information

User Groups
Password Policy Configuration Options
ALTER USERGROUP Statement (Access Control) [page 665]
DROP USERGROUP Statement (Access Control) [page 987]
GRANT Statement (Access Control) [page 1010]
USERGROUPS System View [page 1696]
USERS System View [page 1698]
CREATE USER Statement (Access Control) [page 893]
USERGROUP_PARAMETERS System View [page 1697]

4.10.1.79 CREATE VIEW Statement (Data Definition)

Creates a view on the database.

Syntax

```sql
CREATE [ OR REPLACE ] VIEW <view_name> [ COMMENT <string_literal> ]
 [ ( <column_name_list> ) ] [ ( <parameterized_view_clause> ) ]
 [ WITH EXPRESSION MACROS( <expression_macro_list> ) ]
 [ WITH <annotations> ]
 [ WITH STRUCTURED PRIVILEGE CHECK ]
 [ WITH ANONYMIZATION ( ALGORITHM <algorithm_name> ) [ [ <view_level_parameters> ] ]
```
Syntax Elements

**OR REPLACE**

Replaces the view if it already exists; otherwise, if the view does not exist, it is created.

A CREATE OR REPLACE operation does not change the ID of an existing view, and maintains any privileges associated with the view.

**view_name**

Creates the specified view, with an optional schema name.

```
<view_name> ::= [ <schema_name>.] <identifier>
```

**COMMENT string_literal**

Specifies a descriptive comment for the view.

Specifying this clause saves you from having to execute a separate COMMENT ON statement for the view after it is created.

**column_list**

Specifies the column names for the view.

```
<column_list> ::= <column_name>, <column_name> [, ...]
```

When a column name is specified along with the view name, a query result is displayed with that column name. If a column name is omitted, then a query result gives an appropriate name to the column automatically. The number of column names must be the same as the number of columns returned from `<subquery>`.

**parameterized_view_clause**

Defines the view columns by using parameters that can take input when queried.

```
<parameterized_view_clause> ::= <parameter_definition>, ...
<parameter_definition> ::= IN <parameter_name> <parameter_datatype> [ <default_value> ]
```

<parameterized_view_clause> can be specified with or without a `<column_name_list>` specification, as shown respectively below:

```
CREATE VIEW myView2 (c1, c2) (IN p1 int, IN p2 VARCHAR(10)) AS SELECT col1, col2 FROM myTable1 WHERE col1=:p1 AND col2=:p2;
CREATE VIEW myView1 (IN p1 INT, IN p2 VARCHAR(10)) AS SELECT col1, col2 FROM myTable1 WHERE col1=:p1 AND col2=:p2;
```

The parameter references in the WHERE clause must be preceded by a colon (for example, `:p1`) to differentiate the parameter reference from a column reference.
Parameterizing a view allows you to pass values to search for as parameters when querying the view. For example, `SELECT * FROM v1(1, 'abc');` is semantically equivalent to `SELECT * FROM (SELECT col1, col2 FROM myTable1 WHERE col1=1 AND col2='abc') v1;`.

<default_value> specifies the value to use, based on the <parameter_datatype>, if no value is specified.

**as_subquery**

The query that forms the definition for the data in the view.

```
<as_subquery> ::= ( AS ( <subquery> ) | <with_clause> )
<with_clause> ::= ( WITH <alias> AS ( <subquery> AS <alias> [, ... ] ) SELECT <expression> FROM <alias>
```

**with_clause**

Defines the columns for the view by using a common table expression.

**with_association_clause**

Defines a relationship between the view and one or more tables (or views, in the case of JOIN).

```
<with_association_clause> ::= WITH ASSOCIATIONS ( <association_def_list> )
<association_def_list> ::= <association_def> ... <column_def> ... <join_definition> ::= 
  <join_cardinality> JOIN <table_or_view_identifier> [ AS <table_alias> ] ON { 
  <condition> | <column_def> 
  <join_cardinality> ::=   MANY TO ONE
  | MANY TO MANY
  | ONE TO ONE
  | ONE TO MANY
  <column_def> ::= [ [ <schema_name> ] <table_ref> ] <column_name> AS <column_name>
```

**mask_clause**

Adds new masking expression or drops an existing mask expression.

```
<mask_clause> ::= [ ( DEFAULT | SESSION USER ) ] MASK (< column_name> USING <mask_expression> [, ... ] )
<column_name> ::= <identifier>
<mask_expression> ::= <expression>
```

Masking behavior is supported on row and column tables, and SQL row and calculation views. It is not supported on any other type of tables (virtual tables, extended tables, temporary tables etc.) and views (Join/Olap/Hierarchy views).

Only one masking behavior, definer owner-based (DEFAULT MASK) or session user based masking (SESSION USER MASK), is supported on a table or view with masked columns. You can combine both masking behaviors in an object hierarchy. For example, a view with session user based masking can be created on a table with owner-based masking.

If not specified, DEFAULT is the default.

<mask_expression> can be any type of expression, including a user-defined function, that returns the same data type and length as the original column.

For more information on data masking, see the *SAP HANA Security Guide for SAP HANA Platform*. 

**WITH EXPRESSION MACROS (expression_macro_list)**
WITH EXPRESSION MACROS( <expression_macro_list> )

<expression_macro_list> ::= <expression_macro> [, <expression_macro> [,…] ]

<expression_macro> ::=  <expression> AS <alias>

expression_macro_list is a list of expression macros being created on the specified view.

expression_macro

Use <expression_macro> to reference an existing expression macro. When creating a macro expression that references another macro expression, the macro expression that is being referenced must be defined prior to the macro expression that references it.

When using <expression_macro> to reference an existing expression macro, then the AS clause is optional. Use the AS clause to give the new expression macro a different alias, or omit the AS clause to use the alias of the referenced expression macro.

An example for <expression_macro> is `sum_colA AS sum_colB` where `sum_colA` is a predefined expression macro.

expression

<expression> can be any calculation or aggregation function expression on one or more columns in the query. When using <expression> to create a new expression macro, you must use the AS clause to specify an alias for the expression macro. An example for <expression> that is a new expression macro is `AVG(columnA)`.

alias

<alias> is the name given to the expression macro.

WITH annotations

Specifies view-, column-, and parameter-level annotations in the form of key/value pairs. You can reference annotations in subsequent queries to filter results.

<with_annotation_clause> ::= WITH ANNOTATIONS ( { [ <set_view_annotations> ] [ <set_column_annotations> ] [ <set_parameter_annotations> ] } )

<set_view_annotations> ::= <key_set_operation> ...
<set_column_annotations> ::= <column_annotation> [ <column_annotation> [...] ]
<set_parameter_annotations> ::= <parameter_annotation> [ <parameter_annotation> ] ...
<column_annotation> ::= COLUMN <column_ref> <key_set_operation>
<parameter_annotation> ::= PARAMETER <column_ref> <key_set_operation>
<key_set_operation> ::= SET <key_value_pair_list>
<key_value_pair_list> ::= <key_value_pair> [, <key_value_pair> [,... ]
<key_value_pair> ::= <key> == <value>
<column_ref> ::= <identifier>
<key> ::= <string>
<value> ::= <string>

While you must specify annotation on at least one object with the WITH ANNOTATION clause (view, column, or parameter), there is no limit on the number of annotations or types of annotations you can specify.

<key> and <value> represent the annotations you are configuring for the object (view, column, or parameter).
WITH STRUCTURED PRIVILEGE CHECK

Specifies that a user must have an SQL-based analytic privilege to access the view.

**cache_clause**

Controls whether the user can dynamically or statically cache the result of the view and get stale data. A result cache can improve performance for subsequent queries on the view.

```sql
<cache_clause> ::= [ WITH [ <cache_type> ] CACHE [ NAME <cache_name> ]
  [ RETENTION <minute_value> ]
  [ OF <projection_list> ]
  [ FILTER <filter_condition> ]
  [ <location_clause> ]
  [ FORCE ]
```

**cache_type**

Specify whether your result cache is static or dynamic.

```sql
<cache_type> ::= STATIC | DYNAMIC
```

A view has only one type of result cache. The default value is STATIC. Using a static result cache may result in stale data. A dynamic result cache ensures up-to-date data as it is partially or fully refreshed every time you run a query on the result cache.

**minute_value**

Specifies the result cache retention period.

```sql
<minute_value> ::= <unsigned_integer>
```

Specifying a retention period ensures that data does not exceed the specified RETENTION period. Data that exceeds the retention period is refreshed and up-to-date data is returned. Since a dynamic result cache ensures up-to-date data, specifying a retention period of any value other than zero (0) generates an error. Dynamic result caches are only supported for SQL views that are defined as an aggregation on single-column tables.

**projection_list**

For static result caches, allows you to reduce the cached data by specifying a result cached projection list.

```sql
<projection_list> ::= <projection_name> [{, <projection_name>}]
<projection_name> ::= <column_name> | <aggr_type>(<column_name>)
<aggr_type> ::= { SUM | MIN | MAX | COUNT | AVG }
```

If a column that is not part of the projection list is requested or included in the WHERE clause, then the result cache cannot be exploited. You can also direct an aggregation type of a specific column for the SQL view (not supported for calculation view). You specify a result cached projection list, and then the result cache includes aggregated results of that column and returns aggregated results only. This option is not supported for dynamic result caches.

**filter_condition**

Reduces the result cached data by specifying a filter condition.

```sql
<filter_condition> ::=<condition> OR <condition>
| <condition> AND <condition>
| NOT <condition>
```
Only filtered results are cached. Predicates that contain subqueries are not supported.

For static result caches, all forms of non-parameterized query filters are supported. For parameterized query filters, only conjunctive forms are supported. For dynamic result caches, only parameterized query filters on string type columns in conjunctive forms are supported. For example:

```
column = ? [ AND <and_column = ?> ]
```

**location_clause**

Specifies the location where the result cache entry is created.

```
<location_clause> ::= AT [LOCATION] '<hostname>:<port_number>', ...
```

By default, when you execute a query on a static or dynamic result cached view, the optimizer determines the result cache location. In some cases, the same result cache entry is created at multiple indexservers.

**FORFORCE**

For static result caches, forces the addition of a result cache without checking its cachability. This feature is only available for calculation views. This option is not supported for dynamic result caches.

**WITH DDL ONLY**

Prevents users from querying the view and from updating the underlying table.

**WITH READ ONLY** Prevents the user from updating the view.

**WITH ANONYMIZATION clause**
Specifies data anonymization parameters for the view and its columns. You must execute a REFRESH VIEW statement after configuring anonymization parameters.

```sql
WITH ANONYMIZATION ( ALGORITHM <algorithm_name> { [ <view_level_parameters> ] } )
```

**algorithm_name**

Specifies the algorithm to use to anonymize view data.

```sql
<algorithm_name> ::= { 'K-ANONYMITY' | 'DIFFERENTIAL_PRIVACY' | 'L-DIVERSITY' }
```

**view_level_parameters**

Specifies view-level anonymization parameters.

```sql
&view_level_parameters> ::= PARAMETERS <embedded_hierarchy_expression>
```

<embedded_hierarchy_expression> is a string constant containing view-specific anonymization parameters.

**column_level_parameters**

Specifies column-level anonymization parameters.

```sql
<column_level_parameters> ::= <column_spec> [ <column_spec> […] ]
<column_spec> ::= COLUMN <column_name> PARAMETERS
<embedded_hierarchy_expression>
```

<embedded_hierarchy_expression> is a string constant containing column-specific anonymization parameters.

<column_name> must be the name of a column previously specified in <column_name_list>.

For more information, see the topics on data anonymization in the SAP HANA Administration Guide.

### Description

The CREATE VIEW statement effectively creates a virtual table based on the results of an SQL statement. It is not a real table as it does not contain any data.

Update operations on views are supported when the following conditions are met:

- Each column in the view maps to the column of a single table.
- If a column in a view base table has a NOT NULL constraint without a default value, then the column must be included in view columns so that inserts can be performed.
- The view cannot contain an aggregate or analytic function in a SELECT list. For example, the following functions are not allowed:
  - TOP, SET, or DISTINCT operator in a SELECT list
  - an ORDER BY clause.
- The view cannot contain a subquery in a SELECT list
- The view cannot contain a sequence value (CURRVAL, NEXTVAL)
- The view cannot contain a column view as the base view
If the base views or tables are updatable, then a view on the base views or tables can also be updatable if the above conditions are met.

When WITH CHECK OPTION is specified, the WHERE clause cannot contain LOB data type columns, spatial data type columns, or GENERATED ALWAYS columns. When WITH CHECK OPTION is specified, the WHERE clause of the subquery is checked against the values being inserted or updated.

Permissions

To create or replace a view requires one of the following:

- For views owned by self:
  - If you own all underlying objects of the view, no additional privilege is required.
- For views owned by others:
  - To create a view, you need one of the following:
    - CREATE ANY schema level privilege granted on the view.
  - To replace an existing view, you need one of the following:
    - CREATE ANY and ALTER schema level privilege granted on the view.
  - Regardless of view ownership, you require the proper privilege on its base objects referenced by the view.

Examples

WITH CHECK OPTION example:

Create table A.

```
CREATE COLUMN TABLE A (A INT PRIMARY KEY, B INT);
```

Create a view, v, that selects all records from table A.

```
CREATE VIEW v AS SELECT * FROM A;
```

Create a view, v, with WITH CHECK OPTION. When WITH CHECK OPTION is specified, the WHERE clause of the subquery is checked.

```
CREATE VIEW v AS SELECT * FROM A WHERE A > 0 WITH CHECK OPTION;
INSERT INTO v VALUES(1); -- succeeds
INSERT INTO v VALUES(0); -- fails
UPDATE v set a = 0; -- fails
```

Static and dynamic result cache examples:

Create a view, V, with a static result cache retention period of 10 minutes.

```
CREATE VIEW V AS SELECT DISTINCT A, B FROM A
WITH STATIC CACHE RETENTION 10;
```
Restrict the cache entry location for view_a to the location myhost2:00002.

```sql
CREATE VIEW view_a AS SELECT DISTINCT A, B FROM A
    WITH STATIC CACHE
    AT LOCATION 'myhost2:00002';
```

Create a view with a dynamic result cache.

```sql
CREATE VIEW V AS SELECT DISTINCT A, B FROM A
    WITH DYNAMIC CACHE;
```

The cache is refreshed whenever it is queried.

**Masking examples:**

Create the credit_card_mask function and then create a view on the fictional table credit_card_tab that uses the credit_card_mask function to mask sensitive data.

```sql
CREATE FUNCTION credit_card_mask(input VARCHAR(19))
    RETURNS output VARCHAR(19)
    LANGUAGE SQLSCRIPT
    AS temp VARCHAR(19);
BEGIN
    SELECT LEFT(input, 4) || '-XXXX-XXXX-' ||
    RIGHT(input, 4) INTO temp
    FROM SYS.DUMMY;
    output:= temp;
END;
```

```sql
CREATE VIEW credit_card_view
    AS SELECT * FROM credit_card_tab
    WITH MASK (CREDIT_CARD USING credit_card_mask(credit_card));
```

In this example for data masking, execute the following statements to create three users.

```sql
CREATE USER mask_owner PASSWORD Password1234 NO FORCE_FIRST_PASSWORD_CHANGE;
CREATE USER data_owner PASSWORD Password1234 NO FORCE_FIRST_PASSWORD_CHANGE;
CREATE USER end_user PASSWORD Password1234 NO FORCE_FIRST_PASSWORD_CHANGE;
```

Connect as data_owner and create a table called credit_tab and populate it with sensitive data.

```sql
CONNECT data_owner PASSWORD Password1234;
CREATE ROW TABLE credit_tab (Name VARCHAR(20), CREDIT_CARD VARCHAR(19));
INSERT INTO credit_tab VALUES ('John', '1111-1111-1111-1111');
INSERT INTO credit_tab VALUES ('James', '2222-2222-2222-2222');
```

Connect as mask_owner and create the user-defined function credit_mask and grant execute permission on the function to data_owner.

```sql
CONNECT mask_owner PASSWORD Password1234;
CREATE FUNCTION credit_mask(INPUT VARCHAR(19))
    RETURNS OUTPUT VARCHAR(19) LANGUAGE SQLSCRIPT AS
    temp VARCHAR(19);
BEGIN
    SELECT LEFT(INPUT,4) || '-XXXX-XXXX-' ||
    RIGHT(INPUT,4) INTO temp
    FROM SYS.DUMMY;
    OUTPUT := temp;
END;
GRANT EXECUTE ON credit_mask TO data_owner;
```
Connect as data_owner and create the view credit_view using the credit_mask function as the masking expression.

```
CONNECT data_owner PASSWORD Password1234;
CREATE VIEW credit_view AS SELECT * FROM credit_tab
WITH MASK (CREDIT_CARD USING mask_owner.credit_mask(credit_card));
GRANT SELECT ON credit_view TO end_user;
```

Connect as end_user and query credit_view.

```
CONNECT end_user PASSWORD Password1234;
SELECT * FROM data_owner.credit_view;
```

The query returns the following masked results:

<table>
<thead>
<tr>
<th>NAME</th>
<th>CREDIT_CARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1111-XXXX-XXXX-1111</td>
</tr>
<tr>
<td>James</td>
<td>2222-XXXX-XXXX-2222</td>
</tr>
</tbody>
</table>

Connect as data_owner and alter the masking definition for credit_view.

```
CONNECT data_owner PASSWORD Password1234;
ALTER VIEW credit_view AS SELECT name, credit_card FROM credit_tab
WITH MASK (NAME USING 'AAAA', CREDIT_CARD USING 'XXXX');
```

Connect as end_user and query credit_view again.

```
CONNECT end_user PASSWORD Password1234;
SELECT * FROM data_owner.credit_view;
```

The query returns the following masked results:

<table>
<thead>
<tr>
<th>NAME</th>
<th>CREDIT_CARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td>XXXX</td>
</tr>
<tr>
<td>AAAA</td>
<td>XXXX</td>
</tr>
</tbody>
</table>

Connect as data_owner and drop masking for the name and credit_card columns.

```
CONNECT data_owner PASSWORD Password1234;
ALTER VIEW credit_view DROP MASK (NAME, CREDIT_CARD);
```

Connect as end_user and query credit_view.

```
CONNECT end_user PASSWORD Password1234;
SELECT * FROM data_owner.credit_view;
```

Because data masking has been dropped, all data contained in the view is now visible.
Connect as data_owner and mask data in the credit_card column by executing the following statement.

```
CONNECT data_owner password Password1234;
ALTER VIEW credit_view ADD MASK (CREDIT_CARD USING 'XXXX');
```

Connect as end_user and query credit_view.

```
CONNECT end_user PASSWORD Password1234;
SELECT * FROM data_owner.credit_view;
```

The data in the credit_card column is masked in the result set as follows:

<table>
<thead>
<tr>
<th>NAME</th>
<th>CREDIT_CARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>XXXX</td>
</tr>
<tr>
<td>James</td>
<td>XXXX</td>
</tr>
</tbody>
</table>

Connect as data_owner and alter the masking information on the credit card to once again use the credit_mask function.

```
CONNECT data_owner PASSWORD Password1234;
ALTER VIEW credit_view ALTER MASK (CREDIT_CARD USING mask_owner.credit_mask(credit_card));
```

Connect as end_user and query credit_view.

```
CONNECT end_user PASSWORD Password1234;
SELECT * FROM data_owner.credit_view;
```

The data in the result set is once again masked by using the credit_mask function.

<table>
<thead>
<tr>
<th>NAME</th>
<th>CREDIT_CARD</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>1111-XXXX-XXXX-1111</td>
</tr>
<tr>
<td>James</td>
<td>2222-XXXX-XXXX-2222</td>
</tr>
</tbody>
</table>

WITH EXPRESSION MACROS examples:

The following statements create a table, insert data, and create a view with two expression macros (`sum_a` and `count_a`) that calculate the sum and count of column `t1.a`, respectively:

```
CREATE TABLE t1(a INT);
INSERT INTO t1 VALUES(10);
INSERT INTO t1 VALUES(20);
CREATE VIEW v1 AS SELECT * FROM t1 WITH EXPRESSION MACROS(SUM(a) AS sum_a, COUNT(a) AS count_a);
```
The following statements create a view, v1, with two expression macros:

```sql
CREATE TABLE t(x INT, y INT);
CREATE VIEW v1 AS SELECT * FROM t WITH EXPRESSION MACROS (
  SUM(x) AS sum_x,
  AVG(y) AS avg_y
);
```

Now execute the following statements to create a second view, v2, with three expression macros.

```sql
CREATE VIEW v2 AS SELECT * FROM v1 WITH EXPRESSION MACROS (
  sum_x AS sum_x2,
  avg_y,
  COUNT(x) AS count_x
);
```

The first two expression macros are propagated from v1. sum_x is renamed to sum_x2 for v2, whereas avg_y retains its alias. The third expression macro is newly created as count_x.

**WITH ANNOTATIONS example:**

The following example creates a table with annotations and then creates a view that also has annotations:

```sql
CREATE TABLE t1(c1 INT) WITH ANNOTATIONS(
  SET 't_k1' = 't_v1', 't_k2' = 't_v2'
  COLUMN c1 SET 'c_k1' = 'c_v1', 'c_k2' = 'c_v2');
CREATE VIEW v1 AS SELECT * FROM t1 WITH ANNOTATIONS(
  SET 'v_k1' = 'v_v1', 'v_k2' = 'v_v2'
  COLUMN c1 SET 'c_k1' = 'c_v1', 'c_k2' = 'c_v2'
)
```

**WITH DDL ONLY example:**

The following statements show how you to use the WITH DDL ONLY clause to control whether users can query a view:

```sql
CREATE TABLE t1 (a int, b int);
CREATE VIEW v1 AS SELECT * FROM t1 WITH DDL ONLY;
SELECT * FROM v1; -- invalid view name: cannot select the view because it is in DDL-only mode
SELECT IS_DDL_ONLY from VIEWS where VIEW_NAME = 'V1'; -- returns TRUE
CREATE VIEW v2 AS SELECT * FROM v1; -- you can select from the view
```

**WITH ANONYMIZATION example:**

The following statements create a table called `Illness` with sensitive data in a column called `Illness` (illness-related), and populate the table with data:

```sql
CREATE COLUMN TABLE Illness (   -- Sequential Column
  ID BIGINT NOT NULL PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
  -- Identifiers
  Name VARCHAR(10),
  -- Quasi Identifiers (QIDs) (to be generalized)
  Gender VARCHAR(1) NOT NULL,
  City VARCHAR(10) NOT NULL,
  -- Sensitive Data
  Illness VARCHAR(10) NOT NULL);
INSERT INTO ILLNESS VALUES ('Paul', 'M', 'Paris', 'BRONCHITIS');
INSERT INTO ILLNESS VALUES ('Martin', 'M', 'Munich', 'ANGINA');
INSERT INTO ILLNESS VALUES ('Nils', 'M', 'Nice', 'FLU');
INSERT INTO ILLNESS VALUES ('Annika', 'F', 'Munich', 'BROKEN LEG');
```
You can now create a view that anonymizes the data:

```sql
CREATE VIEW Illness_K_Anon (ID, Gender, Location, Illness) AS SELECT ID, Gender, City AS Location, Illness FROM Illness WITH ANONYMIZATION ( ALGORITHM 'K-ANONYMITY' PARAMETERS '{"data_change_strategy": "qualified", "k": 2}' COLUMN ID PARAMETERS '{"is_sequence": true}' COLUMN Gender PARAMETERS '{"is_quasi_identifier":true, "hierarchy": "embedded": [["F"], ["M"]]}' COLUMN Location PARAMETERS '{"is_quasi_identifier":true, "hierarchy": "embedded": [["Paris", "France"], ["Munich", "Germany"], ["Nice", "France"]]}');
```

After you create the anonymized view, you must refresh it to select from it:

```sql
REFRESH VIEW Illness_K_Anon ANONYMIZATION;
```

View the anonymized data in the view:

```sql
SELECT * FROM Illness_K_Anon;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>GENDER</th>
<th>LOCATION</th>
<th>ILLNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>France</td>
<td>BRONCHITIS</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>Germany</td>
<td>ANGINA</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>France</td>
<td>FLU</td>
</tr>
<tr>
<td>4</td>
<td>*</td>
<td>Germany</td>
<td>BROKEN LEG</td>
</tr>
</tbody>
</table>

Related Information

- ALTER VIEW Statement (Data Definition) [page 667]
- VIEW_EXPRESSION_MACROS System View [page 1708]
- EXPRESSION_MACRO Function (Miscellaneous) [page 193]
- Expressions [page 56]
- REFRESH VIEW Statement (Data Definition) [page 1086]
- VIEWS System View [page 1702]
- VIEW_COLUMNS System View [page 1705]
- SAP HANA Security Guide
4.10.1.80 CREATE VIRTUAL FUNCTION Statement (Procedural)

Creates a virtual function.

Syntax

Syntax 1 - Create a virtual function that allows you to employ smart data access on service interfaces exposed by remote systems (both ODBC/non-ODBC).

```
CREATE VIRTUAL FUNCTION <func_name> [(<parameter_clause>)]
    RETURNS <return_table_type> [SQL SECURITY <mode>] [<package_clause>]
    CONFIGURATION <remote_proc_properties> AT <source_name>
```

Syntax 2 - Create a virtual table user-defined function (TUDF) that points to a remote table user-defined function in another SAP HANA on-premise system or in an SAP HANA Cloud, SAP HANA database.

```
CREATE VIRTUAL FUNCTION <func_name> AT <remote_location_clause>
```

Syntax Elements

`func_name`

Specifies the function name with optional schema name.

```
<func_name> ::= [<schema_name>.]<identifier>
<schema_name> ::= <identifier>
```

`parameter_clause`

Specifies input parameters for the virtual function.

```
<parameter_clause> ::= <parameter> [{,<parameter>}...]
```

These input parameters are only supported when creating a virtual function that is used with the syntax element PACKAGE. The syntax element PACKAGE allows the function to pass information to the custom MR.

`parameter`

Defines a function parameter with associated data type, consisting of a name and the data type definition.

```
<parameter> ::= [IN] <param_name> <datatype>
<param_name> ::= <identifier>
<datatype> ::= <sql_type>
```

`sql_type`
Specifies the data type.

\[
\text{<sql_type>} ::= \text{BIGINT} \mid \text{INTEGER} \mid \text{SMALLINT} \mid \text{TINYINT} \mid \text{DATE} \mid \text{TIME} \mid \text{TIMESTAMP} \mid \text{SMALLDECIMAL} \mid \text{DECIMAL} \mid \text{REAL} \mid \text{DOUBLE}
\]

Virtual functions support scalar functions types and primitive SQL types as input.

**return_type**

\[
\text{<return_type>} ::= \text{<return_table_type>} \mid \text{<table_type>}
\]

Table functions must return a table whose type is defined by `<return_type>`. Non-scalar functions are supported.

**return_table_type**

Defines the structure of the returned table data.

\[
\text{<return_table_type>} ::= \text{TABLE} \left( \text{<ret_column_list>} \right)
\]

**ret_column_list**

Defines the list of columns returned from the function.

\[
\text{<ret_column_list>} ::= \text{<ret_column_elem>}[\{, \text{<ret_column_elem>}\}...]
\]

**ret_column_elem**

Defines the name of column element with associated data type.

\[
\text{<ret_column_elem>} ::= \text{<column_name>} \text{<sql_type>}
\]

**table_type**

Defines the output parameters and you can utilize a table type previously defined with the CREATE TYPE command. See CREATE TYPE for details.

\[
\text{<table_type>} ::= \text{<identifier>}
\]

**SQL SECURITY mode**

Specifies the security mode of the table function.

\[
\text{<mode>} ::= \text{DEFINER} \mid \text{INVOKER}
\]

The default value is DEFINER.

**package_clause**
Defines the identifier of the map reduce program that was developed or imported into SAP HANA repository via the SAP HANA Studio plugin.

```<package_clause> ::= PACKAGE <schema>.<jobPackage>
<schema> ::= <identifier>
<jobPackage> ::= <identifier>`

The map reduce program is a collection of artifacts that can be executed on a Hadoop system. The schema name is optional.

**CONFIGURATION remote_proc_properties**

Defines required and optional properties for executing remote procedures.

```<remote_proc_properties> ::= <string_literal>`

The properties have the format of key-value pairs. The target map reduce program can read these key-value pairs. Additional custom properties can also be passed to the remote procedure. The following table lists the user-defined key properties to define a virtual UDF function

<table>
<thead>
<tr>
<th>Property</th>
<th>Optional</th>
<th>Default</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable_remote_cache</td>
<td>optional</td>
<td>false</td>
<td>Instructs SAP HANA to store results of the initial remote query fragment in HDFS and subsequent queries are read directly from the result set.</td>
</tr>
<tr>
<td>remote_cache_validity</td>
<td>optional</td>
<td>3600</td>
<td>Defines the duration of time (in seconds) after which the cache is stale.</td>
</tr>
</tbody>
</table>

The following table lists the required user defined key properties, to define a virtual UDF function that reads HDFS files:

<table>
<thead>
<tr>
<th>Property</th>
<th>Optional</th>
<th>Default</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>hdfs_location</td>
<td></td>
<td></td>
<td>Defines the directory from where the query would read the HDFS file(s). For example, /user/hive/warehouse/test.</td>
</tr>
<tr>
<td>hdfs_input_format</td>
<td>optional</td>
<td>csv</td>
<td>Defines the type of the data source file for the query.</td>
</tr>
<tr>
<td>hdfs_field_delimiter</td>
<td>optional</td>
<td></td>
<td>Defines the character that delimits columns in a file. The default value is a comma.</td>
</tr>
<tr>
<td>date_format</td>
<td></td>
<td>yyyy-MM-dd</td>
<td>Defines the ISO date format of a date typed column in the format.</td>
</tr>
</tbody>
</table>
The following table lists the required user defined key properties to define a virtual UDF function calling a custom map reduce program:

### User-Defined Key Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Optional</th>
<th>Default</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapred_jobchain</td>
<td></td>
<td></td>
<td>Provides a sequence of all <code>&lt;mapred&gt;</code> properties listed below</td>
</tr>
<tr>
<td>mapred_mapper</td>
<td></td>
<td></td>
<td>Specifies the function that responsible for transforming each element in an input list into a new output list. One or more functions can be listed.</td>
</tr>
<tr>
<td>mapred_reducer</td>
<td></td>
<td></td>
<td>Defines the aggregation function of the output list from a mapper. There could be zero or many reducers.</td>
</tr>
<tr>
<td>mapred_combiner</td>
<td></td>
<td></td>
<td>Defines the aggregation of input from a mapper, normally used to help limit the amount of data passed to a <code>&lt;mapred_reducer&gt;</code>.</td>
</tr>
<tr>
<td>mapred_input</td>
<td></td>
<td></td>
<td>Defines the data source. Values can be an input file input or the result of a mapper function. There can be one or many <code>&lt;mapred_input&gt;</code>.</td>
</tr>
<tr>
<td>mapred_input_format</td>
<td></td>
<td></td>
<td>Defines the data input format.</td>
</tr>
<tr>
<td>mapred_output_format</td>
<td>optional</td>
<td>text</td>
<td>Defines the data output format for writing the results to HDFS.</td>
</tr>
<tr>
<td>mapred_field_delimiter</td>
<td>optional</td>
<td>,</td>
<td>Defines the delimiter between columns when generating output files. The default value is a comma.</td>
</tr>
</tbody>
</table>

**AT source_name**
Defines the name of the remote source.

```sql
<source_name> ::= <identifier>
```

**remote_location_clause**

Specifies the location of the remote table user-defined function.

```sql
<remote_location_clause> ::= "<remote_source>"."<database_name>"."<remote_schema_name>"."<remote_function_name>"
```

**Description**

**Syntax 1**

A virtual function allows you to use smart data access on service interfaces exposed by remote systems (both ODBC/non-ODBC). These service interfaces could be a SQLScript Procedure on remote SAP HANA system (ODBC Compliant) or a custom map reduce job executed on a Hadoop system. This feature complements the Virtual Table feature of SDA.

Remote procedure properties are specific to a given service interface on a remote system. It is a delimited list of name value pairs enclosed in quotes. The source name is the name of the remote source created using the `CREATE REMOTE SOURCE` statement. There is no language type specified and the function body is empty. The execution is delegated to the remote side through respective adapter for the given source name. This function is strictly a table UDF and can be invoked part of regular SQL fragment.

From SP08, HANA supports custom map reduce invocation on a Hadoop system. For this reason, a new adapter called hadoop is included along with a remote source that contains all the necessary connectivity information.

**Syntax 2**

A virtual table user-defined function lets you pass parameters to the corresponding remote table user-defined function to retrieve data from the remote database, which is returned as a table. Virtual table user-defined functions are read only and can only be used with `SELECT` statements. They cannot be used with other data manipulation statements such as `INSERT`, `DELETE`, `UPDATE`, `REPLACE`, or `MERGE INTO`.

**Permissions**

This statement requires the `CREATE VIRTUAL FUNCTION` object privilege.

**Example**

The following example shows how to create a remote destination with the adapter type hadoop.

```sql
CREATE REMOTE SOURCE HOSTA1 ADAPTER
```
The following example explains how to create a virtual function with the remote source created in the above example.

```sql
CREATE VIRTUAL FUNCTION word_count() RETURNS TABLE ( word NVARCHAR(60), count INT) PACKAGE "SYSTEM"."WORD_COUNT" CONFIGURATION 'sap.hana.hadoop.mapper=com.sap.hadoop.examples.WordCountMapper; sap.hana.hadoop.reducer=com.sap.hadoop.examples.WordCountReducer; sap.hana.hadoop.input'/path/to/input' AT HOSTA1;
```

The following example explains how to create a virtual function with the remote source created above.

```sql
```

The following example creates virtual function VTUDF1 that points to remote table user-defined function TUDF1.

```sql
CREATE VIRTUAL FUNCTION MYSCHEMA1.VTUDF1 AT "HANA1"."DB1"."MYSCHEMA2"."TUDF1";
```

**Related Information**

CREATE TYPE Statement (Procedural) [page 892]
Create a Virtual Table User-Defined Function
Virtualizing Table User-Defined Functions
4.10.1.81 CREATE VIRTUAL PROCEDURE Statement (Procedural)

Creates a virtual procedure using the specified programming language that allows execution of the procedure body at the specified remote source.

**Syntax**

```
CREATE VIRTUAL PROCEDURE <proc_name> [([parameter_clause])]  
[SQL SECURITY <mode>]  
[<package_clause>] LANGUAGE <lang>  
AT <source_name>  
[AS  
  BEGIN            <procedure_body>    END    ]
```

**Syntax Elements**

- **proc_name**
  Specifies the virtual procedure name, with optional name.

  ```
  <proc_name> ::= [schema_name.]identifier  
  schema_name ::= <unicode_name>
  ```

- **parameter_clause**
  Specifies the input and output parameters for the procedure.

  ```
  <parameter_clause> ::= <parameter> [,(<parameter>)]...
  ```

- **SQL SECURITY mode**
  Specifies the security mode of the virtual procedure.

  ```
  <mode> ::= DEFINER | INVOKER
  ```

  **DEFINER**
  Performs the execution of the virtual procedure with the privileges of the definer of the procedure. DEFINER is the default.
INVOKER

Performs the execution of the virtual procedure with the privileges of the invoker of the procedure.

LANGUAGE

Specifies the programming language that is used in the procedure body.

```sql
LANGUAGE <lang>
<lang> ::= SCALASPARK
```

In this case, SCALASPARK implies that the procedure body will be in the Scala programming language.

package_clause

(Hadoop Integration only) This optional parameter defines the identifier of the virtual package containing the external library files used to execute the Scala code in the procedure body.

```sql
<package_clause> ::= PACKAGE <schema>.<jobPackage>
<schema> ::= <identifier>
<jobPackage> ::= <identifier>
```

AT source_name

Specifies the remote source for which the procedure is created and executed.

procedure_body

Defines the source code for the virtual procedure. Run complex algorithms on both structured (such as tables) and non-structured (such as log files) data using the specified language. Currently, Scala is the only supported language.

Description

Creates a virtual procedure using the specified programming language that allows execution of the procedure body at the specified remote source.

Permissions

This statement requires the CREATE VIRTUAL PROCEDURE object privilege.

Example

```sql
CREATE VIRTUAL PROCEDURE SYSTEM.FINDNGRAMS(
  IN N INT,
  OUT NGRAMS TABLE(STR TEXT)
) LANGUAGE SCALASPARK
AT SPARK_OAKL
AS
BEGIN
  import sqlContext.implicits._
```
import scala.collection.mutable.WrappedArray
import org.apache.spark.ml.feature.NGram

// $example on$
val wordDataFrame = sqlContext.createDataFrame(Seq(
  (0, Array("Hi", "I", "heard", "about", "Spark")),
  (1, Array("I", "wish", "Java", "could", "use", "case", "classes")),
  (2, Array("Logistic", "regression", "models", "are", "neat"))
)).toDF("id", "words")
val ngram = new NGram().setN(N).setInputCol("words").setOutputCol("ngrams")
val ngramDataFrame = ngram.transform(wordDataFrame)
ngramDataFrame.select("ngrams").show(false)

NGRAMS = ngramDataFrame.select("ngrams").map(y=>y(0).asInstanceOf[WrappedArray[_]].mkString",").toDF

END;
CALL FINDNGRAMS(6, ?);

Related Information

VIRTUAL_PACKAGES System View [page 1716]
VIRTUAL_PROCEDURES System View [page 1717]

4.10.1.82 CREATE VIRTUAL TABLE Statement (Data Definition)

Creates a virtual table at a remote source.

Syntax

Syntax 1 - Create a virtual table.

```
CREATE VIRTUAL TABLE <virtual_table_name> AT <remote_location_clause> [ <remote_property_clause> ]
```

Syntax 2 - Create a remote table while creating a virtual table.

```
CREATE VIRTUAL TABLE <virtual_table_name> <table_contents_source>
AT <remote_location_clause> WITH REMOTE
```

Syntax Elements

```
virtual_table_name
```
Specifies a unique name for the virtual table.

```
<virtual_table_name> ::= [ <schema_name>.]<identifier>
```

table_contents_source

Specifies the source from which the table definition is derived.

```
<table_contents_source> ::= ( <table_element> [, <table_element> [, ... ] ] )
```

table_element

Defines the table columns with associated column or table constraints.

```
<table_element> ::= <column_definition> [ <column_constraint> ]
```

column_definition

Defines a table column.

```
<column_definition> ::= <column_name> <data_type> [ <default_value_clause> ]
```

column_name

Specifies the column name.

```
<column_name> ::= <identifier>
```

data_type

Specifies the column data types.

```
<data_type> ::= DATE
    | TIME
    | SECONDDATE
    | TIMESTAMP
    | TINYINT
    | SMALLINT
    | INTEGER
    | BIGINT
    | SMALLDECIMAL
    | REAL
    | DOUBLE
    | VARCHAR [ (<unsigned_integer>) ]
    | NVARCHAR [ (<unsigned_integer>) ]
    | VARBINARY [ (<unsigned_integer>) ]
    | DECIMAL [ (<unsigned_integer> [, <unsigned_integer> ])]
    | FLOAT [ (<unsigned_integer>) ]
    | BOOLEAN
```

default_value_clause

Specifies a value to be assigned to the column if an INSERT statement does not provide a value for the column.

```
<default_value_clause> ::= DEFAULT <default_value_exp>
<default_value_exp> ::= NULL
    | <string_literal>
    | <signed_numeric_literal> <unsigned_numeric_literal>
    | <datetime_value_function>
<datetime_value_function> ::= CURRENT_DATE
column_constraint

Specifies the column constraint rules.

\[
\texttt{column\_constraint} ::= \texttt{NULL} \mid \texttt{NOT NULL} \mid \texttt{<unique\_specification>}
\]

**NULL**

Allows NULL values in the column. If NULL is specified it is not considered a constraint, it represents that a column that may contain a null value. The default is NULL.

**NOT NULL**

Prohibits NULL values in the column.

**unique_specification**

Specifies unique constraints.

\[
\texttt{unique\_specification} ::= \texttt{UNIQUE} \mid \texttt{PRIMARY KEY}
\]

**UNIQUE**

Specifies a column as a unique key. A composite unique key enables the specification of multiple columns as a unique key. With a unique constraint, multiple rows cannot have the same value in the same column.

A UNIQUE column can contain multiple NULL values.

**PRIMARY KEY**

Specifies a primary key constraint, which is a combination of a NOT NULL constraint and a UNIQUE constraint.

remote_location_clause

Identifies a remote object (table or view) at an existing remote source.

\[
\texttt{remote\_location\_clause} ::= \texttt{"<remote\_source\_name>"."<database\_name>"."<schema\_name>"."<identifier>"}
\]

For SAP HANA, SAP IQ, SAP MaxDB, Terradata, and IBM DB2 remote sources, you can use NULL for \texttt{<database\_name>},

remote_property_clause

Specifies the parameters and their values to be set for the virtual table as XML.

\[
\texttt{remote\_property\_clause} ::= \\
\texttt{[REMOTE PROPERTY 'dataprovisioning\_parameters'=<dataprov\_params\_value>]}
\texttt{<dataprov\_params\_value} ::= \texttt{<string\_literal>}
\]

The REMOTE PROPERTY clause allows you to specify data provisioning parameters for the virtual table that is being created. The parameters list is sent to the adapter as a parameter for the remote object metadata request.

**WITH REMOTE** Creates a table on the remote source using the \texttt{<table\_contents\_source>} definition and a corresponding virtual table on the local source. The WITH REMOTE option is supported for remote
sources using the following SDA adapters: hanaodbc, iqodbc, aseodbc, tdodbc, voraodbc, odbc (Oracle, Microsoft SQL Server, IBM Netezza, IBM DB2, BigQuery). It is not supported with an SDI adapter.

Description

The CREATE VIRTUAL TABLE provides a way to access an existing table/view on a remote source from an SAP HANA instance. The list of remote columns is automatically imported into the virtual table.

Use the DROP TABLE `<table_name>` to drop a virtual table.

Permissions

Syntax 1 - This syntax requires the CREATE VIRTUAL TABLE object privilege.

Syntax 2 - This syntax requires the CREATE VIRTUAL TABLE and REMOTE TABLE ADMIN object privileges. If the credential type of the remote source is PASSWORD, then only the owner of the remote sources who set the technical user credentials has the authorization to use the WITH REMOTE clause. For all other credential types, authorization is a function of the privileges of the user on the remote source.

Examples

Create virtual table VT on existing remote source S. The originating table is `tableA`, which is within the schema SYSTEM on HA1.

```
CREATE VIRTUAL TABLE VT AT "S"."HA1"."SYSTEM"."tableA";
```

Create a virtual table where the following string is sent to the adapter as a parameter for the metadata request: `<Parameter name="objects">1,10,1000,300</Parameter>|<Parameter name="stateHierarchy">USA</Parameter>|<Parameter name="date">2014-06-01</Parameter>`

```
CREATE VIRTUAL TABLE ...     REMOTE PROPERTY 'dataprovisioning_parameters'='<Parameter name="objects">1,10,1000,300</Parameter>    <Parameter name="stateHierarchy">USA</Parameter><Parameter name="date">2014-06-01</Parameter>';
```

Create table A1 on remote source Remote1 and create a corresponding virtual table virtual_A1 in the local source.

```
CREATE VIRTUAL TABLE virtual_A1 (a int, b int) at "Remote1"."HA1"."SYSTEM"."A1" WITH REMOTE;
```
4.10.1.83 CREATE WORKLOAD CLASS Statement (Workload Management)

Defines workload classes.

Syntax

CREATE WORKLOAD CLASS <workload_class_name> [ <inheritance> ] [ <property_list> ] [ ENABLE | DISABLE ]

Syntax Elements

**workload_class_name**

Creates the specified workload class.

<workload_class_name> ::= <identifier>

**inheritance**

Specifies a parent workload class.

<inheritance> ::= PARENT { <parent_name> | NULL }

The workload class inherits the TOTAL STATEMENT MEMORY LIMIT or TOTAL STATEMENT THREAD LIMIT value from the parent workload class.

The child workload class must have the corresponding individual limit properties of the parent workload class and must not have aggregated properties, for example, a workload class with class-wise limits like total statement thread and memory limit.

The parent workload class must have a TOTAL STATEMENT MEMORY LIMIT or TOTAL STATEMENT THREAD LIMIT property and no individual properties, for example statement thread/memory limit.

Hierarchies can be only a single level (parent-child). Specifying NULL unsets the parent workload class.

**property_list**

Defines the properties of a workload class. The properties have the format of key-value pairs.

property_list ::= [SET | UNSET] { '<key>' = '<value>' }[,'<key>' = '<value>' ] [ ...]
PRIORITY Specifies a value from 0-9. A higher number specifies a higher priority.

STATEMENT MEMORY LIMIT Specifies a limit in GB. For example, STATEMENT MEMORY LIMIT = '2' specifies a 2 GB limit.

STATEMENT THREAD LIMIT Specifies a statement concurrency limit, rather than a thread limit; SAP HANA does not impose a thread limit on statements.

STATEMENT TIMEOUT Specifies an expiry time for statement execution, after which statement execution is canceled and the transaction is rolled back. The default is 0 (no expiry).

TOTAL STATEMENT MEMORY LIMIT Specifies an overall total memory limit, in GB, for all statements belonging to a specific workload class. If this property is specified, then PRIORITY and TOTAL STATEMENT THREAD LIMIT must also be specified.

TOTAL STATEMENT THREAD LIMIT Specifies an overall limit for the number of active threads scheduled for statements for the specific workload class. If this property is specified, then PRIORITY and TOTAL STATEMENT MEMORY LIMIT must also be specified.

WRITE TRANSACTION LIFETIME Specifies the duration of uncommitted write transactions, in minutes, before the connection is terminated. If this value is set to 0, then the timeout behavior is disabled.

IDLE CURSOR LIFETIME Specifies the duration of cursors, in minutes, before the connection is terminated. If this value is set to 0, then the timeout behavior is disabled.

ADMISSION CONTROL REJECT CPU THRESHOLD / ADMISSION CONTROL REJECT MEMORY THRESHOLD Specifies the threshold value as a percentage to reject a request based on CPU or memory consumption, if the measured load is equal to or larger than the configured value from the workload class. The threshold range is 0 - 100, where 0 rejects all requests and 100 accepts all requests. This property only applies to new incoming requests (statements already running are not affected).

A request rejection returns the message 'rejected by workload class configuration'.

Precedence is workload class, then session-wise admission control.

ADMISSION CONTROL QUEUE CPU THRESHOLD / ADMISSION CONTROL QUEUE MEMORY THRESHOLD Specifies the threshold range as a value based on CPU or memory consumption. The threshold range is 0 - 100, where value 0 represents always queuing and value 100 represents no queuing. If
the measured load is equal to or larger than configured queueing threshold from workload class, an arrived request is enqueued. If the waiting time in the queue of a certain request greater than the queue_timeout value, the issued request is rejected.

A request rejection returns the message 'queue wait timeout exceeded'.

This property only handles new incoming requests (already running statements are not affected) as with session-wise admission control.

Precedence is REJECTION comparison, then QUEUEING comparison. Workload class takes precedence over admission control.

**ENABLE or DISABLE**

Enables or disables the workload class.

## Description

A workload class with only NULL values behaves like the default workload class.

An aggregated workload class must have all properties specified (TOTAL STATEMENT MEMORY LIMIT, TOTAL STATEMENT THREAD LIMIT, PRIORITY).

**Precedence in the case of competing settings:** In the case of competing settings between workload class settings and user parameter settings (set by using ALTER USER), a workload class setting takes precedence over a user parameter setting. Workload class settings and user parameter settings always take precedence over .ini file settings. If the STATEMENT TIMEOUT setting, which can also be set by the client, is set to something more restrictive by the client than the value that is determined after evaluating the setting on the server side, then the more restrictive client-side setting is used instead. The following table demonstrates how precedence is determined:

<table>
<thead>
<tr>
<th>global.ini</th>
<th>User parameters</th>
<th>Workload class</th>
<th>Resulting effective values (statement execution)</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5</td>
<td>if not matched class found</td>
<td>priority 5, thread limit 5, memory limit 50GB</td>
</tr>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5, memory 5</td>
<td>priority 7, thread limit 10, memory limit 30GB</td>
<td>priority 7, thread limit 10, memory limit 30GB</td>
</tr>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5, memory 30GB</td>
<td>priority 7, thread limit 10, memory limit undefined</td>
<td>priority 7, thread limit 10, memory limit 30GB</td>
</tr>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5, memory 30GB</td>
<td>priority 7, total thread limit 10, total memory limit 30GB</td>
<td>priority 7, total thread limit 10, total memory limit 30GB</td>
</tr>
<tr>
<td>memory 50 GB</td>
<td>priority 5, thread 5, memory 30GB</td>
<td>priority 7, total thread limit undefined, total memory limit 50GB (SQL error returned at CREATE/ALTER WORKLOAD CLASS)</td>
<td>priority 5, thread limit 5, memory limit 30GB</td>
</tr>
</tbody>
</table>
### User parameters

<table>
<thead>
<tr>
<th></th>
<th>memory 50 GB</th>
<th>priority 5, thread 5, memory 30 GB</th>
<th>priority 7, total thread limit 10, total memory limit undefined (SQL error returned at CREATE/ALTER WORKLOAD CLASS)</th>
<th>priority 5, thread limit 5, memory limit 30GB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>statement timeout</strong></td>
<td>2 seconds</td>
<td>N/A</td>
<td>5 seconds</td>
<td>statement time out is 5 unless the value was set more restrictively by the client. For example, if the client-side setting is 3, then 3 becomes the effective value.</td>
</tr>
</tbody>
</table>

### Permissions

The use of this statement requires the WORKLOAD_ADMIN privilege.

### Examples

The following example defines a new workload class called MyWorkloadClass with all three properties defined.

```sql
CREATE WORKLOAD CLASS "MyWorkloadClass"
    SET 'PRIORITY' = '3', 'STATEMENT MEMORY LIMIT' = '2', 'STATEMENT THREAD LIMIT' = '20';
```

The following example defines new workload classes and mappings and demonstrates how to enable and disable them.

```sql
CREATE WORKLOAD MAPPING "MyMapping1" WORKLOAD CLASS "MyClass" SET 'CLIENT'='001';
CREATE WORKLOAD MAPPING "MyMapping2"
    WORKLOAD CLASS "MyClass" SET 'CLIENT'='002';
ALTER WORKLOAD CLASS "MyClass" DISABLE; -- disables "MyClass" and its mappings,
    "MyMapping1" and "MyMapping2"
ALTER WORKLOAD CLASS "MyClass" ENABLE; -- enables "MyClass" and its mappings,
    "MyMapping1" and "MyMapping2"
CREATE WORKLOAD CLASS "MyClass1"
    SET 'PRIORITY' = '0',
        'STATEMENT MEMORY LIMIT' = '3',
        'STATEMENT THREAD LIMIT' = '10';
CREATE WORKLOAD MAPPING "MyMapping3"
    WORKLOAD CLASS "MyClass1" SET 'CLIENT'='003';
ALTER WORKLOAD CLASS ALL DISABLE; -- disables all workload classes, "MyClass" and "MyClass1", and their mappings ("MyMapping1", "MyMapping2" and "MyMapping3")
ALTER WORKLOAD CLASS ALL ENABLE; -- enables all workload classes and related mappings again
```
The following example creates a class called `class1` and sets the overall memory threshold to 30 GB and the overall thread limit to 50 for every statement that belongs to `class1`.

```sql
CREATE WORKLOAD CLASS class1 SET 'TOTAL STATEMENT MEMORY LIMIT'='30', 'TOTAL STATEMENT THREAD LIMIT'='50', 'PRIORITY' = '5';
```

This example creates a workload class called `class1` with a write transaction lifetime of 10 minutes and an idle cursor lifetime of 11 minutes.

```sql
CREATE WORKLOAD CLASS class1 SET 'WRITE TRANSACTION LIFETIME'='10', 'IDLE CURSOR LIFETIME' = '11';
```

This example creates a workload class called `class2` which rejects all requests (threshold = 0).

```sql
CREATE WORKLOAD CLASS class2 SET 'ADMISSION CONTROL REJECT CPU THRESHOLD' = '0', 'ADMISSION CONTROL REJECT MEMORY THRESHOLD' = '0';
```

This example creates a workload class called `class3` which accepts all requests; no rejections (threshold = 100).

```sql
CREATE WORKLOAD CLASS class3 SET 'ADMISSION CONTROL REJECT CPU THRESHOLD' = '100', 'ADMISSION CONTROL REJECT MEMORY THRESHOLD' = '100';
```

This example creates a workload class called `class4` with a CPU threshold of 70 and a memory threshold of 80.

```sql
CREATE WORKLOAD CLASS class4 SET 'ADMISSION CONTROL REJECT CPU THRESHOLD' = '70', 'ADMISSION CONTROL REJECT MEMORY THRESHOLD' = '80';
```

This example creates two workload classes, PARENT_CLS and CHILD_CLS. CHILD_CLS inherits the total statement memory limit from PARENT_CLS.

```sql
CREATE WORKLOAD CLASS "PARENT_CLS" SET 'TOTAL STATEMENT MEMORY LIMIT' = '40';
CREATE WORKLOAD CLASS "CHILD_CLS" PARENT "PARENT_CLS" SET 'STATEMENT MEMORY LIMIT' = '10';
```

### Related Information

- ALTER WORKLOAD CLASS Statement (Workload Management) [page 678]
- WORKLOAD_CLASSES System View [page 1721]
4.10.1.84 CREATE WORKLOAD MAPPING Statement
(Workload Management)

Defines workload mappings.

Syntax

```
CREATE WORKLOAD MAPPING <mapping_name>
WORKLOAD CLASS <workloadclass_name> [ <property_list> [ <wildcard_option> ] ]
```

Syntax Elements

**mapping_name**

Specifies the name for the new workload mapping.

```
<mapping_name> ::= <identifier>
```

**workloadclass_name**

Specifies the workload class name.

```
<workloadclass_name> ::= <identifier>
```

**property_list**

Defines the workload mapping properties.

```
<property_list> ::=   { [ SET <key_value_pair_list> ] | [ UNSET <key_value_pair_list> ] }  
<key_value_pair_list> ::= <key_value_pair> [, <key_value_pair> [,... ] ]  
<key_value_pair> ::= '<key>'='<value>'
```

**<key> ::=**

APPLICATION USER NAME
| CLIENT
| APPLICATION COMPONENT NAME
| APPLICATION COMPONENT TYPE
| APPLICATION NAME
| {USER NAME | USERGROUP NAME}
| SCHEMA NAME
| OBJECT NAME
| XS APPLICATION USER NAME
| APPLICATION SOURCE

**<value> ::= <string_literal>**

USERGROUP NAME and USER NAME cannot be configured together. For established connections, changes to USERGROUP NAME are only applied when a connected database client reconnects. If there are two matched workload classes by USER NAME and USERGROUP NAME respectively, then USER NAME takes precedence over USERGROUP NAME.
SCHEMA NAME and OBJECT NAME must be configured together. SCHEMA NAME is the schema of the object. OBJECT NAME can be a procedure name, application function library (AFL) area or package name.

If an AFL area or package name is specified, then the workload class is applied to all AFLLANG procedures that call AFs in that AFL area.

If there are multiple workload classes matched by SCHEMA NAME and OBJECT NAME, then the following order of precedence is used applied: AREA --> PACKAGE --> PROCEDURE.

XS APPLICATION USER NAME takes precedence over APPLICATION USER NAME.

APPLICATION SOURCE value must be set with setCommandInfo() method for it to be considered when mapping to a workload class.

**wildcard_option**

Specifies the wildcard character to use for value matching.

```sql
<wildcard_option> ::= WITH WILDCARD [ <wildcard-character> ]
<wildcard-character> ::= <ASCII_character>
```

Wildcard characters can only be ASCII characters and cannot be used with USER NAME or USERGROUP NAME, SCHEMA NAME, or OBJECT NAME.

Only prefix wildcards are supported, for example WILDCARD %.

If no wildcard character is specified, then the default wildcard character is %. Use one wildcard character per property.

**Permissions**

You need the WORKLOAD_ADMIN system privilege.

**Description**

This statement defines workload mappings.

Workload mapping settings are stored in the WORKLOAD_MAPPINGS system view.

**Examples**

The following statement creates an entry for the MyWorkloadMapping1 workload mapping for the ABCADM database user and the BW application.

```sql
CREATE WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyWorkloadClass"
SET 'USER NAME' = 'ABCADM', 'APPLICATION NAME' = 'BW';
```
The following statement uses the wildcard character % by default for value matching any application name that begins with BW.

```
CREATE WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyWorkloadClass"
SET 'APPLICATION NAME' = 'BW%' WITH WILDCARD;
```

The following two statements both result in an error since the wildcard must be placed at the end of the property value.

```
CREATE WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyWorkloadClass"
SET 'APPLICATION NAME' = 'BW%123' WITH WILDCARD;
```

```
CREATE WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyWorkloadClass"
SET 'APPLICATION NAME' = 'BW%*123' WITH WILDCARD '*';
```

The following statement creates an entry for the MyWorkloadMapping1 workload mapping for the USERGROUP_TEST user group.

```
CREATE WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyWorkloadClass"
SET 'USERGROUP NAME' = 'USERGROUP_TEST';
```

The following statement creates an entry for the "MyWorkloadMapping1" workload mapping for the object SYS.CHECK_ES.

```
CREATE WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "DATAMART" SET
'SCHEMA NAME' = 'SYS', 'OBJECT NAME' = 'CHECK_ES';
```

The following statement creates an entry for the MyWorkloadMapping1 workload mapping for the XS application user name TESTER.

```
CREATE WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "DATAMART" SET 'XS APPLICATION USER NAME' = 'TESTER';
```

The following statement uses the wildcard character % by default for any XS application user name that begins with ADMIN.

```
CREATE WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "DATAMART" SET 'XS APPLICATION USER NAME' = 'ADMIN%' WITH WILDCARD;
```

The following statement creates an entry for the MyWorkloadMapping1 workload mapping where the APPLICATION SOURCE value is exactly "example.cc:123".

```
CREATE WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyClass1" SET
'APPLICATION SOURCE' = 'example.cc:123';
```

The following statement uses the wildcard character % for any APPLICATION SOURCE value that begins with "example.cc:"

```
CREATE WORKLOAD MAPPING "MyWorkloadMapping1" WORKLOAD CLASS "MyClass1" SET
'APPLICATION SOURCE' = 'example.cc:%' WITH WILDCARD '%';
```
4.10.1.85  CREATE X509 PROVIDER (Access Control)

Defines an X.509 provider in the SAP HANA database.

Syntax

```
CREATE X509 PROVIDER <x509_provider_name> <issuer_clause> [ <matching_rules_clause> ]
```

Syntax Elements

**x509_provider_name**

Specifies the identifier of an X.509 provider to be created. The X.509 provider name must not already be defined.

```
<x509_provider_name> ::= <simple_identifier>
```

**issuer_clause**

Sets the issuer distinguished name for the X.509 provider.

```
<issuer_clause> ::= WITH ISSUER <issuer_distinguished_name>  
<issuer_distinguished_name> ::= <string_literal>
```

**matching_rules_clause**

Specifies one or more rules for matching external identities to database users. A matching rule is a distinguished name where one attribute has a value of ‘*’. This attribute contains the user name that needs to be matched during logon. All other attributes must match and be in the same order.

Matching rules are tried in the same order they are defined in the provider, and the first one that matches is used.

```
<matching_rules_clause> ::= MATCHING RULES  
'<'subject_distinguished_name_mapping>'  
[', 'subject_distinguished_name_mapping>' ... ]  
<subject_distinguished_name_mapping> ::= <string_literal>
```
Permissions

Only database users with the USER ADMIN system privilege can create an X.509 provider.

Examples

Create an X.509 provider.

```
CREATE X509 PROVIDER MyProvider WITH ISSUER 'CN=DigiCert Global Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US';
```

Create an X.509 provider with a matching rule.

```
CREATE X509 PROVIDER MyProvider WITH ISSUER 'CN=DigiCert Global Root CA, OU=www.digicert.com, O=DigiCert Inc, C=US'
MATCHING RULES 'CN=*, OU=SAP SE, C=DE';
```

Related Information

X.509 Certificate-Based User Authentication
ALTER X509 PROVIDER (Access Control) [page 684]
DROP X509 PROVIDER (Access Control) [page 991]
X509_PROVIDERS System View [page 1725]
X509_PROVIDER_RULES System View [page 1725]
X509_USER_MAPPINGS System View [page 1726]

4.10.1.86 DELETE Statement (Data Manipulation)

Deletes records from a table where a specified condition is met.

Syntax

```
DELETE [HISTORY] FROM <table_name> [ <partition_restriction> ]
[ FOR PORTION OF APPLICATION_TIME FROM <from_value> TO <to_value> ]
[ WHERE <condition> ] [ <hint_clause> ]
```
Syntax Elements

**HISTORY**
Marks the selected records of a history table for deletion.

**FROM table_name**
Specifies the table name where the deletion is to occur, with optional schema name.

```sql
<table_name> ::= [[:<database_name>.[]<schema_name>.[]<identifier>]
<schema_name> ::= <unicode_name>
```

For linked database, `<database_name>` is the name of the remote source. `<identifier>` is the name of the table on the remote source.

**partition_restriction**
For a partitioned table, `<partition_restriction>` specifies the partition in which the target rows are located.

```sql
<partition_restriction> ::= PARTITION ( <partition_number> )
<partition_number> ::= <unsigned_integer>
```

If you do not specify a partition restriction, then the database checks all partitions for the delete. But if there is a partition restriction, then only the rows in the specified partition are deleted. Use the TABLE_PARTITIONS system view to determine the partition number.

The `<partition_restriction>` clause is supported for multistore tables but not extended store tables.

**WHERE condition**
Specifies the conditions where the command should be performed.

```sql
<condition> ::= <condition> OR <condition>
          | <condition> AND <condition>
          | NOT <condition>
          | ( <condition> )
          | <predicate>
          | CURRENT OF <cursor>
```

WHERE CURRENT OF `<cursor>` specifies to perform the delete at the current position in the cursor.

For more information on `<predicate>`, see the Predicates topic in this guide.

**hint_clause**
Specifies the hint clause. For information on specifying hints, see the HINT clause of the SELECT statement.

**FOR PORTION OF APPLICATION_TIME** Deletes rows that are fully contained within the given time period. FROM and TO values for partially overlapping rows will be adjusted accordingly.

**Description**

If the WHERE clause is omitted, then DELETE removes all records from a table.
When using the DELETE HISTORY command, time travel queries referencing the deleted rows may still access these rows. To physically delete these rows, you must execute the following statements:

```
ALTER SYSTEM RECLAIM VERSION SPACE;
MERGE HISTORY DELTA of <table_name>;
```

In some cases even the execution of the two statements above may not lead to physical deletion.

To check whether the rows are physically deleted, execute the following statement:

```
SELECT * FROM <table_name> WHERE <condition>
   WITH PARAMETERS ('REQUEST_FLAGS'= ('ALLCOMMITTED','HISTORYONLY'));
```

The "WITH PARAMETERS ('REQUEST_FLAGS'= ('ALLCOMMITTED','HISTORYONLY'))" clause is only supported for validating the result of using the DELETE HISTORY statement.

**Examples**

This example deletes values equal to 1 from table T.

Create a table T and insert some data.

```
CREATE ROW TABLE T (KEY INT PRIMARY KEY, VAL INT);
INSERT INTO T VALUES (1, 1);
INSERT INTO T VALUES (2, 2);
INSERT INTO T VALUES (3, 3);
```

Delete from table T where the key column is equal to 1:

```
DELETE FROM T WHERE KEY = 1;
```

After executing the above query, the contents of table T are as follows, showing that one row was deleted from the table:

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

This example deletes partition 1 from table T1.

```
CREATE COLUMN TABLE T1 (a int, b int) PARTITION BY RANGE (a)  
(PARTITION 0 <= VALUES < 10, PARTITION OTHERS);
INSERT INTO T1 PARTITION(1) VALUES(1,50);
DELETE FROM T1 PARTITION(1);
```

This example deletes the updated record history.

Create a history table and insert a value.

```
CREATE HISTORY COLUMN TABLE HIST(KEY INT PRIMARY KEY, VAL INT);
INSERT INTO HIST VALUES (1, 1);
COMMIT;
```
Update the row you inserted in the previous step.

```
UPDATE HIST SET VAL=42 WHERE KEY=1;
COMMIT;
```

Merge in the table delta-storage with the table main-storage.

```
MERGE DELTA OF HIST;
```

Assuming the merge statement has moved the updated record into the history storage of the history table, delete the updated record history.

```
DELETE HISTORY FROM HIST WHERE KEY=1;
```

Delete from the table using an array value.

```
DELETE FROM T1 WHERE 3 MEMBER OF C1;
```

The following statements provide an example of updating and deleting multiple tables (currently only COLUMN tables are supported) through an updatable cursor. In this case, you must specify columns of tables to be locked using FOR UPDATE OF within the cursor’s SELECT statement. DML execution using updatable cursor is allowed just one time per row.

```
CREATE COLUMN TABLE employees (employee_id INTEGER, employee_name VARCHAR(30),
department_id INTEGER);
INSERT INTO employees VALUES (1, 'John', 1);
INSERT INTO employees VALUES (2, 'Sam', 2);
INSERT INTO employees VALUES (3, 'Julie', 3);
INSERT INTO employees VALUES (4, 'Kate', 4);
CREATE COLUMN TABLE departments (department_id INTEGER, department_name VARCHAR(20));
INSERT INTO departments VALUES (1, 'Development');
INSERT INTO departments VALUES (2, 'Operation');
INSERT INTO departments VALUES (3, 'HR');
INSERT INTO departments VALUES (4, 'Security');
DO BEGIN
  DECLARE CURSOR cur FOR SELECT employees.employee_name,
departments.department_name
  FROM employees, departments WHERE employees.department_id =
departments.department_id
  FOR UPDATE OF employees.employee_id, departments.department_id;
  FOR r AS cur DO
    IF r.department_name = 'Development' THEN
      UPDATE employees SET employee_id = employee_id + 10000, department_id =
department_id + 100
      WHERE CURRENT OF cur;
      UPDATE departments SET department_id = department_id + 100
      WHERE CURRENT OF cur;
    ELSEIF r.department_name = 'HR' THEN
      DELETE FROM employees WHERE CURRENT OF cur;
      DELETE FROM departments WHERE CURRENT OF cur;
    END IF;
  END FOR;
END;
```

This example creates a table, populates it with data and then deletes rows fully contained within the date range '2018-8-4' to '2018-8-8'. The VALIDFROM and VALIDTO values of the partially overlapping rows are updated accordingly.

```
CREATE COLUMN TABLE T1(VALIDFROM DATE NOT NULL, VALIDTO DATE NOT NULL, PKEY INT,
VALUE INT , PERIOD FOR APPLICATION_TIME(VALIDFROM, VALIDTO));
```
INSERT INTO T1 VALUES('2018-8-1', '2018-8-3', 1, 9494);
INSERT INTO T1 VALUES('2018-8-3', '2018-8-5', 1, 20);
INSERT INTO T1 VALUES('2018-8-5', '2018-8-7', 1, 30);
INSERT INTO T1 VALUES('2018-8-7', '2018-8-9', 1, 40);
delete from t1 for portion of application_time from '2018-8-4' to '2018-8-8'
where pkey = 1;
select * from t1 order by validfrom, validto;
-- ('2018-8-1', '2018-8-3', 1, 9494) -- untouched row
-- ('2018-8-3', '2018-8-4', 1, 20) -- validto value updated accordingly for
partially overlapping row
-- ('2018-8-8', '2018-8-9', 1, 40) -- validfrom value updated accordingly for
partially overlapping row

DELETE FROM T1 FOR PORTION OF APPLICATION_TIME FROM '2018-8-4' TO '2018-8-8'
WHERE PKEY = 1;
SELECT * FROM T1 ORDER BY VALIDFROM, VALIDTO;
-- ('2018-8-1', '2018-8-3', 1, 9494) - Row is unchanged
-- ('2018-8-3', '2018-8-4', 1, 20) - VALIDTO value is updated accordingly for
partially overlapping row
-- ('2018-8-8', '2018-8-9', 1, 40) - VALIDFROM value updated accordingly for
partially overlapping row

Related Information

TABLE_PARTITIONS System View [page 1681]
Introduction to SQL [page 27]
Predicates [page 61]
SELECT Statement (Data Manipulation) [page 1104]
HINTS System View [page 1585]
HINT Details [page 1134]

4.10.1.87 DO BEGIN...END Statement (Procedural)

Executes an anonymous block a single time.

Syntax

DO [ [ <named_parameters_list> ] ] [ ( <bound_parameter_clause> ) ]
BEGIN [ <block_properties> ]
<variable_declaration>
<statement_list>
Syntax Elements

**named_parameters_list**

Specifies the list of named parameters.

\[
\text{<named_parameters_list>} := ([ \text{<named_parameter>}] [\{ ,\text{<named_parameter>}\} ... ])
\]

**named_parameter**

Specifies the name of the parameter.

\[
\text{<named_parameter>} := \text{<parameter_mode>}, \text{<parameter_name>}, \text{<type_name>}
\]

**bound_parameter_clause**

Specifies the input and output parameters for the procedure with value bindings.

\[
\text{<bound_parameter_clause>} := \text{<bound_parameter>} [, ... ]
\]

**bound_parameter**

Specifies a procedure parameter with its associated data type and value binding.

\[
\text{<bound_parameter>} := \text{<parameter>} \Rightarrow \text{<proc_param>}
\]

**proc_param**

Specifies procedure parameters.

**block_properties**

Specifies one or more properties to apply to the block.

\[
\text{<block_properties>} :=
\begin{align*}
\text{SEQUENTIAL EXECUTION} \\
\text{PARALLEL EXECUTION} \\
\text{AUTONOMOUS TRANSACTION}
\end{align*}
\]

**SEQUENTIAL EXECUTION**

Specify SEQUENTIAL EXECUTION to sequence the execution of independent DML statements and read-write procedure calls.

**PARALLEL EXECUTION**

Specify PARALLEL EXECUTION to parallelize the execution of independent DML statements and read-write procedure calls.

**AUTONOMOUS TRANSACTION**

Specify AUTONOMOUS TRANSACTION if you want the block to be executed independently of the main procedure. Changes made and committed using AUTONOMOUS TRANSACTION persist regardless of whether the main procedure transaction is committed or rolled back. The end of an AUTONOMOUS TRANSACTION block has an implicit commit.

**variable_declaration**
Declares variables for use in anonymous block.

```plaintext
<variable_declaration> ::= 
  <declaration> 
  [ <declaration> ]  
  [...]
<declaration> ::= 
  DECLARE <variable_name> { <sql_data_type> | TABLE ( <column_definition> [,..] ] 
  | DECLARE <variable_name> AUTO <default_value>;
<column_definition> ::= <column_name> <sql_data_type>
<default_value> ::= { ( DEFAULT | := | = ) { <value> | <expression> }
```

Specifying AUTO allows you to declare a variable and its default value without having to explicitly set the data type.

**Statement list**

Specifies one or more statements to execute as part of the anonymous block.

**Description**

An anonymous block is only executed once. All SQLScript statements supported in procedures are also supported in anonymous blocks. Unlike procedures, an anonymous block has no corresponding object created in the catalog.

Anonymous blocks are defined and executed in a single step. Therefore, lifecycle handling like CREATE or DROP is not needed. The body of an anonymous block is similar to the procedure body.

To return a result set, use a SELECT statement because anonymous blocks do not have any parameters defined.

For output parameters, `?` is a valid value and cannot be omitted because otherwise the query parameter cannot be bound. For input parameters, the name of a physical table can be set. For scalar input parameters, any scalar expression can be used, except for scalar UDF. DEFAULT EMPTY and INOUT parameters are not supported.

**Examples**

Execute an anonymous block that creates a table and inserts values into that table.

```plaintext
DO BEGIN
  DECLARE I INTEGER;
  CREATE ROW TABLE TAB1 (I INTEGER);
  FOR I IN 1..10 DO
      INSERT INTO TAB1 VALUES (:I);
  END;
END;
```

Execute an anonymous block and return the result set by using a SELECT statement.

```plaintext
DO BEGIN
```

---

SAP HANA SQL Reference Guide for SAP HANA Platform

SQL Reference
T1 = SELECT I, 10 AS J FROM TAB;
T2 = SELECT I, 20 AS K FROM TAB;
T3 = SELECT J, K FROM :T1, :T2 WHERE :T1.I = :T2.I;
SELECT * FROM :T3;
END;

Execute an anonymous block that calls another procedure.

DO BEGIN
   T1 = SELECT * FROM TAB;
   CALL PROC3(:T1, :T2);
   SELECT * FROM :T2;
END;

Execute an anonymous block that includes an exception handler.

DO BEGIN
   DECLARE I, J INTEGER;
   BEGIN
      DECLARE EXIT HANDLER FOR SQLEXCEPTION
      IF ::SQL_ERROR_CODE = 288 THEN
         DROP TABLE TAB;
         CREATE ROW TABLE TAB (I INTEGER PRIMARY KEY);
      ELSE
         RESIGNAL;
      END IF;
      CREATE ROW TABLE TAB (I INTEGER PRIMARY KEY);
      FOR I in 1..3 DO
         INSERT INTO TAB VALUES (:I);
      END FOR;
      IF :J <> 3 THEN
         SIGNAL SQL_ERROR_CODE 10001;
      END IF;
   END;
END;

Execute an anonymous block that uses COMMIT and ROLLBACK.

DO BEGIN
   CREATE ROW TABLE TAB2 (K INT);
   COMMIT;
   DROP TABLE TAB;
   CREATE ROW TABLE TAB (J INT);
   ROLLBACK;
   DELETE FROM TAB;
END;

Execute an anonymous block that calls a procedure to process the selected data.

DO BEGIN
   T1 = SELECT I, 10 AS J FROM TAB;
   T2 = SELECT I, 20 AS K FROM TAB;
   T3 = SELECT J, K FROM :T1, :T2 WHERE :T1.I = :T2.I;
   CALL PROC3(:T3, :T4);
   SELECT * FROM :T4;
END;

Execute an anonymous block that runs a loop to insert data into a table.

DO
BEGIN
    DECLARE I INTEGER;
    FOR I in 1..3 DO
        INSERT INTO TAB VALUES (:I);
    END FOR;
END;

Execute an anonymous block that computes the sum of two integers.

DO (IN A INT => 1, IN B INT => ?, OUT C INT => ?) BEGIN
    C = :A + :B;
END;

Related Information

Anonymous Block
Explicit Parallel Execution
CREATE PROCEDURE Statement (Procedural) [page 776]

4.10.1.88 DROP AUDIT POLICY Statement (Access Control)

Drops an audit policy.

Syntax

DROP AUDIT POLICY <policy_name> [ FOR <database_name> ]

Syntax Elements

policy_name

Specifies the name of the audit policy to be dropped.

<policy_name> ::= <identifier>

<policy_name> must specify an existing audit policy.
database_name Defines the name of the tenant to drop the audit policy to. This option can only be used on SYSTEMDB.
Description

Even if an audit policy is dropped, the audit action specified in the dropped audit policy can be audited further. This happens if another audit policy is enabled that specifies the dropped audit action.

To temporarily stop auditing an audit policy, disable it.

Since there is no automatic deletion of audit log entries for deleted policies, audit log entries created by such a policy need to be cleaned up manually.

Permissions

To drop an audit policy on SYSTEMDB or on a tenant, you must have the AUDIT ADMIN privilege.

To use the FOR <database_name> option, you need the DATABASE AUDIT ADMIN privilege granted on SYSTEMDB, not on the tenant.

Example

Create the priv_audit audit policy.

```
CREATE AUDIT POLICY priv_audit AUDITING SUCCESSFUL GRANT PRIVILEGE, REVOKE PRIVILEGE, GRANT ROLE, REVOKE ROLE LEVEL CRITICAL;
```

Drop the priv_audit audit policy.

```
DROP AUDIT POLICY priv_audit;
```

Drop the priv_audit1 audit policy on SYSTEMDB for tenant HA2.

```
DROP AUDIT POLICY priv_audit1 FOR HA2;
```

Related Information

AUDIT_POLICIES System View [page 1503]
4.10.1.89 DROP CERTIFICATE Statement (System Management)

Drops a specified certificate from the list of certificates.

Syntax

```sql
DROP CERTIFICATE { <certificate_name> | <certificate_id> }
```

Syntax Elements

- **certificate_name**
  
  Specifies the name of the certificate to be dropped.
  
  `<certificate_name> ::= <identifier>`

- **certificate_id**
  
  Specifies the certificate to be dropped.
  
  `<certificate_id> ::= <identifier>`

Description

The DROP CERTIFICATE statement drops this certificate from the list of certificates, which can be assigned to a PSE store. The statement succeeds only if the certificate is not assigned to a PSE store.

Information about certificates that are usable for assignment to PSE stores is stored in the CERTIFICATES system view.

Permissions

Only users with the CERTIFICATE ADMIN system privilege are allowed to drop certificates.
Examples

Assuming the certificate name is SAP_CERT, drop it.

```
DROP CERTIFICATE SAP_CERT;
```

Related Information

CERTIFICATES System View [page 1516]

4.10.1.90 DROP CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption)

Drops a column encryption key (CEK) and any key copies for it from the schema.

Syntax

```
DROP CLIENTSIDE ENCRYPTION COLUMN KEY [ <schema>.]<key_name>
```

Syntax Elements

- `schema_name`
  
  Specifies the schema name.

```
<schema_name> ::= <unicode_name>
```

- `key_name`
  
  Specifies the name of the CEK to drop.

Description

When you execute this statement, the CEK and all its key copies are dropped from the catalog.

If there are columns encrypted with the CEK, then the operation fails and nothing is dropped. All columns using the column encryption key must first be dropped or must stop using the CEK.
Permissions

This statement requires the CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege.

Examples

The following statement drops the column encryption key `<hrapp_cek1>`:

```sql
DROP CLIENTSIDE ENCRYPTION COLUMN KEY mychema.hrapp_cek1;
```

Related Information

- CLIENTSIDE_ENCRYPTION_COLUMN_KEYS System View [page 1517]
- CLIENTSIDE_ENCRYPTION_KEYPAIRS System View [page 1519]
- CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 741]
- ALTER CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 435]
- DROP CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 949]
- CREATE CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 743]
- DROP CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 950]

4.10.1.91 DROP CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption)

Drops a client key pair (CKP).

Syntax

```sql
DROP CLIENTSIDE ENCRYPTION KEYPAIR <keypair_name>
```

Syntax Elements

- `keypair_name`
  
  Specifies the name of the CKP.
### Description

Executing this statement removes the specified CKP from the catalog along with every column encryption key copy assigned to it. The client driver and the SAP HANA server collaborate transparently to remove this key pair from the client hdbkeystore.

The SAP HANA server ensures that at least one other key administrator key copy (CEK copy encrypted by CKP of user with CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege) exists for the column encryption key being dropped. If there are no key administrator key copies, then the statement fails.

### Permissions

Users can drop key pairs they created. However, this statement requires the DROP CLIENTSIDE ENCRYPTION KEYPAIR privilege to drop key pairs that were created by other users.

### Examples

The following example drops the client key pair `user1_ckp`:

```
DROP CLIENTSIDE ENCRYPTION KEYPAIR user1_ckp;
```

### Related Information

- [CLIENTSIDE_ENCRYPTION_COLUMN_KEYS System View](page 1517)
- [CLIENTSIDE_ENCRYPTION_KEYPAIRS System View](page 1519)
- [CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption)](page 741)
- [ALTER CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption)](page 435)
- [DROP CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption)](page 949)
- [CREATE CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption)](page 743)
- [hdbkeystore Utility]
4.10.1.92 DROP COLLECTION Statement (JSON Document Store)

Drops a collection table.

Syntax

```sql
DROP COLLECTION [ TABLE ] <collection_name> <drop_option>
```

Syntax Elements

- **collection_name**
  Specifies the name of the collection table to drop.

  ```sql
  <collection_name> ::= [[<database_name>.]<schema_name>.]<identifier>
  ```

- **drop_option**
  Specifies the drop option to use.

  ```sql
  <drop_option> ::= CASCADE | RESTRICT
  ```

  CASCADE drops the collection table and any dependent objects. RESTRICT drops the collection table only when dependent objects do not exist. If RESTRICT is specified and a dependent object exists, then an error is returned.

  If `<drop_option>` is not specified, then a non-cascaded drop is performed, which only drops the specified collection table. Dependent objects of the collection table are invalidated but not dropped.

Description

Drops a collection table from the database. When you drop a collection table, all JSON documents stored in the collection table are removed as well.

DROP COLLECTION fails if `<identifier>` is a regular table, not a collection table.

Examples

The following statement drops a collection table called `MyCollection`.

```sql
DROP COLLECTION MyCollection;
```
The following statement drops a collection table called `MyCollection2` and any associated views.

```
DROP COLLECTION MyCollection CASCADE;
```

### 4.10.1.93 DROP CREDENTIAL Statement (Access Control)

Drops an existing component-specific or application-specific credential.

**Syntax**

```
DROP CREDENTIAL FOR [USER <user_name>] COMPONENT <component_id>
    PURPOSE <purpose_def> TYPE <type_def>
```

**Syntax Elements**

- **user_name**
  Specifies the owner of the credential to be dropped.
  
  `<user_name> ::= <unicode_name>`

  Only users with the CREDENTIAL ADMIN system privilege can drop credentials for other users.

- **component_id**
  Specifies an identifier for the component that uses the credential.

  `<component_id> ::= <string_literal>`

- **purpose_def**
  Specifies the application-specific or component-specific purpose string.

  `<purpose_def> ::= <string_literal>`

- **type_def**
  Specifies a connection mechanism; for example 'PASSWORD'.

  `<type_def> ::= <string_literal>`

**Description**

Each user can drop credentials that they own.
Permissions

This statement requires the CREDENTIAL ADMIN privilege.

Example

Create a credential for user WORKER that is used by the component INTERNAL_APP. This credential uses the password mechanism for the PURPOSE COMPANY_MASTER_MACHINE.

```
CREATE CREDENTIAL FOR USER WORKER COMPONENT 'INTERNAL_APP' PURPOSE 'COMPANY_MASTER_MACHINE' TYPE 'PASSWORD' USING 'PASSWORD_9876';
```

Drop the credential.

```
DROP CREDENTIAL FOR USER WORKER COMPONENT 'INTERNAL_APP' PURPOSE 'COMPANY_MASTER_MACHINE' TYPE 'PASSWORD';
```

Related Information

CREDENTIALS System View [page 1524]

4.10.1.94 DROP DATABASE Statement (Tenant Database Management)

Drops a tenant database.

Syntax

```
DROP DATABASE <database_name> [ DROP BACKUPS ]
```

Syntax Elements

`database_name`

Specifies the database name.

```
<database_name> ::= <identifier>
```
**DROP BACKUPS**

Drops all current backup directories for the tenant database. Backup directories that were previously in use, and backups that are written to third-party backup tools, are not deleted.

**Description**

DROP DATABASE drops the specified tenant database.

**Permissions**

The use of this statement requires the DATABASE ADMIN privilege.

**Example**

The following example shows you how to drop the my_database database.

```
DROP DATABASE my_database;
```

**Related Information**

[SAP HANA Database Backup and Recovery](#)

**4.10.1.95 DROP FULLTEXT INDEX Statement (Data Definition)**

Removes a full text index.

**Syntax**

```
DROP FULLTEXT INDEX <index_name>
```
Syntax Elements

index_name

Drops the specified full text index.

\[ \text{index_name} ::= \text{[schema_name].identifier} \]

Description

The DROP FULLTEXT INDEX statement removes a full text index.

Example

Create table A and then add an index, i, on column b of table A.

```
CREATE ROW TABLE A (a INT, b NVARCHAR(10), c NVARCHAR(20));
CREATE INDEX i ON a(b);
```

Drop index i.

```
DROP FULLTEXT INDEX i ON a(b);
```

4.10.1.96 DROP FUNCTION Statement (Procedural)

Deletes a function from the database.

Syntax

```
DROP FUNCTION <func_name> [drop_option]
```

Syntax Elements

func_name

Specifies the name of the function to be dropped, with optional schema name.

\[ \text{func_name} ::= \text{[schema_name].identifier} \]
<schema_name> ::= <unicode_name>

drop_option

Specifies whether a cascaded drop is performed.

<drop_option> ::= CASCADE | RESTRICT

When <drop_option> is not specified, a non-cascaded drop is performed. This only drops the specified function; dependent objects of the function are invalidated but not dropped.

The invalidated objects can be revalidated when an object that has same schema and object name is created.

    CASCADE
        Drops the function and dependent objects.
    RESTRICT
        Drops the function only when dependent objects do not exist. If this drop option is used and a dependent object exists an error is thrown.

Description

Drops a function created using CREATE FUNCTION from the database catalog.

Examples

Drop a function called my_func from the database using a non-cascaded drop.

    DROP FUNCTION my_func;

4.10.1.97  DROP GRAPH WORKSPACE Statement (Data Definition)

Drops a graph workspace.

Syntax

    DROP GRAPH WORKSPACE <workspace_name>
Syntax Elements

workspace_name

Drops the specified graph workspace.

\[
<\text{workspace\_name}> ::= [<\text{schema\_name}>.]<\text{identifier}>
\]

Description

Drops a graph workspace.

4.10.1.98 DROP INDEX Statement (Data Definition)

Removes an index.

Syntax

\[
\text{DROP INDEX } <\text{index\_name}> \ [\ \text{ONLINE} \ [\ \text{PREFERRED} \ ] \ ]
\]

Syntax Elements

index_name

Drops the specified index, with the optional schema name.

\[
<\text{index\_name}> ::= [<\text{schema\_name}>.]<\text{identifier}>
\]

\[
<\text{schema\_name}> ::= <\text{unicode\_name}>
\]

ONLINE [ PREFERRED ]

Allows the dropping of the index without serializing with concurrent DML operations. ONLINE is only supported for column-store, non-replicated tables. The drop results in an operation that only acquires a shared table lock.

When PREFERRED is specified, if ONLINE execution is not supported, then a SQL warning message appears, and the drop index operation executes in table x-lock mode.
Description

The DROP INDEX statement removes an index.

Example

Create table A and then add an index, i, on column b of table A.

CREATE TABLE A (a INT, b NVARCHAR(10), c NVARCHAR(20));
CREATE INDEX i ON A(b);

Drop index i.

DROP INDEX i;

Related Information

INDEXES System View [page 1585]
INDEX_COLUMNS System View [page 1587]

4.10.1.99 DROP JWT PROVIDER Statement (Access Control)

Drops a JWT provider in the SAP HANA database.

Syntax

DROP JWT PROVIDER <jwt_provider_name> [ CASCADE ]

Syntax Elements

jwt_provider_name

Specifies which JWT provider to drop.

<jwt_provider_name> ::= <simple_identifier>

<jwt_provider_name> must be an existing JWT provider.
CASCADE Required only when the JWT provider is referenced by a PSE’s purpose object. The provider is removed from any PSE referencing it along with any user mappings for users authenticating with the provider. For PSEs where it is the only provider assigned, the purpose of that PSE is also removed.

Description

If the JWT provider is in use by an SAP HANA database user, then the CASCADE parameter is required to successfully drop the JWT provider.

Permissions

Only database users that have the USER ADMIN system privilege can drop a JWT provider.

Examples

Drops the JWT provider, my_jwt_provider1.

```
DROP JWT PROVIDER my_jwt_provider1;
```

Drops the JWT provider, my_jwt_provider2, which is referenced by the PSE mypse1.

```
DROP JWT PROVIDER my_jwt_provider2 CASCADE;
```

Related Information

CREATE JWT PROVIDER Statement (Access Control) [page 767]
ALTER JWT PROVIDER Statement (Access Control) [page 455]
JWT_PROVIDERS System View [page 1589]
JWT_USER_MAPPINGS System View [page 1591]
4.10.1.100  DROP LIBRARY Statement (SQLScript)

Drops a SQLScript user-defined library.

**Syntax**

```
DROP LIBRARY [ <schema_name>.]<library_name>
```

**Syntax Elements**

- `library_name`
  - Specifies the name of the SQLScript user-defined library to drop.

**Description**

For more information on SQLScript user-defined libraries, see the SAP HANA SQLScript Reference.

**Examples**

The following example drops the mylib library:

```
DROP LIBRARY mylib;
```

**Related Information**

- User-Defined Libraries
- SAP HANA SQLScript Reference
- CREATE LIBRARY Statement (SQLScript) [page 770]
- ALTER LIBRARY Statement (SQLScript) [page 462]
- LIBRARIES System View [page 1600]
- LIBRARY_MEMBERS System View [page 1601]
4.10.1.101 DROP LDAP PROVIDER Statement (Access Control)

Drops an LDAP provider, and its associated credential, from the internal secure credential store.

Syntax

DROP LDAP PROVIDER <ldap_provider_name>

Syntax Elements

ldap_provider_name
The identifier for a valid LDAP provider previously created using the CREATE LDAP PROVIDER statement.

<ldap_provider_name> ::= <unicode_name>

Description

Drops an LDAP provider.

If the dropped LDAP provider is the default LDAP provider, subsequent connections for users configured for LDAP authorization are impacted as follows:

• Any user connections created before the expiration of the value for the ldap_authorization_role_reuse_duration database property will succeed.
• Any user connections created after the expiration of the value for the ldap_authorization_role_reuse_duration database property will fail.
• Any first time user connections will fail.

With regards to LDAP authentication, if the default LDAP provider is dropped, subsequent user connections fail.

Permissions

Only users with the LDAP ADMIN privilege can drop LDAP providers.
Examples

The following example drops an LDAP provider called `my_ldap_provider`.

```
DROP LDAP PROVIDER my_ldap_provider;
```

Related Information

- CREATE LDAP PROVIDER Statement (Access Control) [page 772]
- ALTER LDAP PROVIDER Statement (Access Control) [page 458]
- VALIDATE LDAP PROVIDER Statement (Access Control) [page 1195]
- LDAP_PROVIDER_URLS System View [page 1599]
- LDAP_PROVIDERS System View [page 1597]
- LDAP_USERS System View [page 1599]

4.10.1.102 DROP PROCEDURE Statement (Procedural)

Deletes a procedure from the database.

Syntax

```
DROP PROCEDURE <proc_name> [DROP <drop_option>]
```

Syntax Elements

- **proc_name**
  
  Specifies the name of the procedure to be dropped, and optionally, a schema name.

  ```
  <proc_name> ::= [<schema_name>.]<identifier>
  <schema_name> ::= <unicode_name>
  ```

- **drop_option**
  
  Specifies whether a cascaded drop is performed.

  ```
  <drop_option> ::= CASCADE | RESTRICT
  ```

  When `<drop_option>` is not specified, then a non-cascaded drop be performed. Only the specified procedure is dropped; dependent objects of the procedure are invalidated, but they are not dropped.
The invalidated objects can be revalidated when an object that has same schema and object name is created.

**CASCADE**

Drops the procedure and dependent objects.

**RESTRICT**

Drops the procedure only when dependent objects do not exist. If this drop option is used and a dependent object exists, then an error is thrown.

**Description**

Drops a procedure created using CREATE PROCEDURE from the database catalog.

**Examples**

Drop a procedure called my_proc from the database using a non-cascaded drop.

```
DROP PROCEDURE my_proc;
```

### 4.10.1.103 DROP PSE Statement (System Management)

Drops a PSE store.

**Syntax**

```
DROP PSE <pse_name>
```

**Syntax Elements**

- **pse_name**

  Specifies the name of the PSE store to drop.

  ```
  <pse_name> ::= <identifier>
  ```
Description

Dropping of a PSE store fails if the PSE store is assigned to a purpose. Existing certificates remain in the Certificate Store.

When the owner of a PSE Store is dropped, their PSE Stores are dropped as well, even if they are assigned to a purpose. So for example, with SSL enforced, dropping a certain user may render the whole HDB unusable.

Permissions

To use this statement, you must have the DROP privilege on the PSE Store object.

Examples

Drop the PSE store examplepse.

```
DROP PSE examplepse;
```

4.10.1.104 DROP PUBLIC KEY Statement (System Management)

Drops a public key from the list of keys. The key cannot be dropped if it is used in a PSE store.

Syntax

```
DROP PUBLIC KEY  <public_key_name>
```

Syntax Elements

```
public_key_name
```

Specifies the name of the public key to drop.

```
<public_key_name> ::= <identifier>
```
Permissions

To use this statement, you must have the CERTIFICATE ADMIN system privilege.

Examples

Drop the public key.

```
DROP PUBLIC KEY jwt_pubkey;
```

4.10.1.105 DROP REMOTE SOURCE Statement (Access Control)

Removes an existing remote source.

Syntax

```
DROP REMOTE SOURCE <remote_source_name> [<drop_option>]
```

Syntax Elements

**remote_source_name**

Specifies the identifier of the remote source.

```
<remote_source_name> ::= <identifier>
```

**drop_option**

Specifies the drop option to use.

```
<drop_option> ::= CASCADE | RESTRICT
```

When `<drop_option>` is not specified, a RESTRICT drop is performed.

CASCADE drops the remote source and dependent objects.

RESTRICT drops the remote source only when dependent objects do not exist. If this drop option is used and a dependent object exists, then an error is thrown.
**Description**

The DROP REMOTE SOURCE statement removes an existing remote source.

**Permissions**

This statement requires the DROP object privilege on the remote source.

**Example**

Create the remote source S and then drop it.

```
CREATE REMOTE SOURCE S ADAPTER ASEODBC
  CONFIGURATION 'DSN=ProductionASE'
  WITH CREDENTIAL TYPE 'PASSWORD' USING
    'user="aseuser";password="asepassword"';
DROP REMOTE SOURCE S;
```

4.10.1.106  DROP ROLE Statement (Access Control)

Drops a role.

**Syntax**

```
DROP ROLE <role_name>
```

**Syntax Elements**

`role_name`

Drops the specified role, with the optional schema name.

```
<role_name> ::= [<schema_name>].<identifier>
<schema_name> ::= <unicode_name>
```

<role_name> must specify an existing role.
Description

Roles that are delivered with the SAP HANA database cannot be dropped: PUBLIC, CONTENT_ADMIN, MODELING, MONITORING, and SAP_INTERNAL_HANA_SUPPORT.

If a role has been granted to a user or another role, then it is revoked when the role is dropped. Revoking a role may lead to making some views inaccessible or making procedures not executable, which occurs if a view or procedures depends on any of the privileges of the dropped role.

Note that when dropping a role mapped to an LDAP group, in addition to revoking the role from users to whom it was granted, its mapping to LDAP group is deleted.

Permissions

Only database users with the ROLE ADMIN system privilege can drop a role. Each user that has this privilege is allowed to drop any role.

Example

Create a role with the name role_for_work_on_my_schema.

```
CREATE ROLE role_for_work_on_my_schema;
```

Drop the role_for_work_on_my_schema role.

```
DROP ROLE role_for_work_on_my_schema;
```

Related Information

ALTER ROLE Statement (Access Control) [page 475]
CREATE ROLE Statement (Access Control) [page 802]
ROLES System View [page 1646]
GRANTED_ROLES System View [page 1581]
GRANTED_PRIVILEGES System View [page 1580]
4.10.1.107 DROP SAML PROVIDER Statement (Access Control)

Drops the specified SAML provider.

Syntax

```
DROP SAML PROVIDER <saml_provider_name> CASCADE
```

Syntax Elements

- **saml_provider_name**
  Specifies the name of the SAML provider to drop.
  
  ```
  <saml_provider_name> ::= <simple_identifier>
  ```

  - `<saml_provider_name>` must be an existing SAML provider.

- **CASCADE**
  Required only when the SAML purpose is referenced by a PSE's purpose object. The provider is removed from any PSE referencing it along with any user mappings for users authenticating with the provider. For PSEs where it is the only provider assigned, the purpose of that PSE is also removed.

Description

If the SAML provider is currently used by an SAP HANA database user, then the CASCADE parameter is required to successfully drop the SAML provider.

Permissions

Only database users with the USER ADMIN system privilege can drop a SAML provider.

Example

Drop the SAML provider named ac_saml_provider1.

```
DROP SAML PROVIDER ac_saml_provider1;
```
Drops the SAML provider, ac_saml_provider2, which is referenced by the PSE mypse1.

```
DROP SAML PROVIDER ac_saml_provider2 CASCADE;
```

**Related Information**

- PSE_PURPOSE_OBJECTS System View [page 1621]
- SAML_PROVIDERS System View [page 1648]

### 4.10.1.108 DROP SCHEDULER JOB Statement (Data Definition)

Drops a scheduled job in the current or specified schema.

**Syntax**

```
DROP SCHEDULER JOB [ <schema_name>.] <job_name>
```

**Syntax Elements**

- `<job_name>` Specifies the name of the scheduled job being dropped.

**Permissions**

This statement requires that you own the scheduled job or have the DROP object privilege on the scheduled job.

**Example**

This example drops the scheduler job schedule1

```
DROP SCHEDULER JOB schedule1;
```
4.10.1.109 DROP SCHEMA Statement (Data Definition)

Removes a schema.

Syntax

```
DROP SCHEMA <schema_name> [drop_option]
```

Syntax Elements

- **schema_name**
  
  Drops the specified schema.

  ```
  <schema_name> ::= <unicode_name>
  ```

- **drop_option**
  
  Specifies the drop option to use.

  ```
  <drop_option> ::= CASCADE | RESTRICT
  ```

  CASCADE drops the schema and all objects contained in the schema, regardless of which user created them. If there are objects in another schema that refer to objects in the schema being dropped, they are dropped if they belong to the user who is dropping the schema, otherwise they are invalidated.

  Invalidated objects can be re-validated when an object that has same schema name is created.

  RESTRICT drops the schema, but only when there are no objects in it. If RESTRICT is specified while there are still objects in the schema, then an error is returned.

  If <drop_option> is not specified, then RESTRICT is used by default.

  Invalidated objects can be re-validated when an object that has same schema name is created.

Description

The DROP SCHEMA statement removes a schema.

Dropping a schema also drops any expression macros defined for the view. The list of all expression macros for a schema can be obtained by querying the VIEW_EXPRESSION_MACROS system view.

If you drop a schema, any schema synonym that references it is also dropped.
Example

Create a schema named my_schema and a table named my_schema.t.

```sql
CREATE SCHEMA my_schema;
CREATE ROW TABLE my_schema.t (a INT);
```

Drop my_schema with a CASCADE option.

```sql
DROP SCHEMA my_schema CASCADE;
```

### 4.10.1.110 DROP SCHEMA SYNONYM Statement (Data Definition)

Removes a schema synonym.

#### Syntax

```sql
DROP SCHEMA SYNONYM <synonym_name>
```

#### Syntax Elements

- **synonym_name** The name of the schema synonym to drop.

#### Description

Use DROP SCHEMA SYNONYM to remove a schema synonym.

#### Permissions

You must have the CREATE SCHEMA system privilege.

A schema synonym is automatically dropped when the schema to which it refers is dropped.
Example

Execute the following statement to create a schema synonym called TEST_SCHEMA_SYN that references the schema SYSTEM:

```sql
CREATE OR REPLACE SCHEMA SYNONYM TEST_SCHEMA_SYN FOR SYSTEM;
```

Execute the next statement to drop that same schema synonym:

```sql
DROP SCHEMA SYNONYM TEST_SCHEMA_SYN;
```

Related Information

- DROP SCHEMA Statement (Data Definition) [page 971]
- CREATE SCHEMA Statement (Data Definition) [page 807]
- CREATE SCHEMA SYNONYM Statement (Data Definition) [page 808]
- SET SCHEMA Statement (Session Management) [page 1170]

4.10.1.111 DROP SEQUENCE Statement (Data Definition)

Removes a sequence.

Syntax

```sql
DROP SEQUENCE <sequence_name> [ <drop_option> ]
```

Syntax Elements

- `sequence_name`
  
  Drops the specified sequence, with the optional schema name.

  ```sql
  <sequence_name> ::= [ <schema_name> . ] <identifier>
  ```

- `drop_option`
  
  Specifies the drop option to use.

  ```sql
  <drop_option> ::= CASCADE | RESTRICT
  ```
CASCADE drops the sequence and dependent objects. RESTRICT drops the sequence only when dependent objects do not exist. If this drop option is used and a dependent object exists, then an error is thrown.

If `<drop_option>` is not specified, then a non-cascaded drop is performed, which only drops the specified sequence. Dependent objects of the sequence are invalidated but not dropped.

**Description**

The `DROP SEQUENCE` statement removes a sequence.

Invalidated objects can be re-validated when an object that has same schema and object name is created.

**Example**

Create a sequence named `seq`.

```
CREATE SEQUENCE seq START WITH 11;
```

Drop the sequence named `seq`.

```
DROP SEQUENCE seq;
```

**4.10.1.112 DROP STATISTICS Statement (Data Definition)**

Drops user-defined data statistic objects that the query optimizer uses to make decisions for query plans.

**Syntax**

```
DROP STATISTICS { <data_statistics_name>[.<data_statistics_name>[,...] ] 
| ON <data_sources> [ [HAVING] <match_properties> ] };
```

**Syntax Element**

`data_statistics_name`

Specifies the name of the data statistics object.

```
<data_statistics_name> ::= [schemas_name].<identifier>
```
<schema_name> ::= <identifier>

data_sources
Specifies the data source(s) of the data statistics objects.

<data_sources> ::= <table_name> [ ( <column_name>[, <column_name>[,...] ] ) [ <match_type> ]

table_name
Specifies the table on which the data statistics are defined.

<table_name> ::= [ [ <database_name>.]<schema_name>.]<identifier>

For linked database, <database_name> is the name of the remote source. For all other cases, <database_name> is the name of the database where the table is located.

column_name
Specifies the column for which the data statistics are defined.

<column_name> ::= <identifier>

If no <column_name> is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

match_type
Controls which data statistics objects to match to <data_sources>.

<match_type> ::= EXACT | CASCADE

If <match_type> is not specified, then any data statistics object(s) that reference all or some of the columns, but no other columns specified in <data_sources> are refreshed.

Specify EXACT to refresh data statistics objects that precisely match <data_sources> (including column order).

Specify CASCADE to refresh data statistics objects that reference at least one column in <data_sources>.

Use this table to understand how matching is performed based on <match_type> when <data_sources> is T(A, B, C):

<table>
<thead>
<tr>
<th>Match type</th>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>(not specified)</td>
<td>T(A, C)</td>
<td>T(A, X) - because T(X) is not a column in &lt;data_sources&gt;.</td>
</tr>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B, A)</td>
<td></td>
</tr>
<tr>
<td>EXACT</td>
<td>T(A, B, C)</td>
<td>T(B, A, C) - because the column order is different than the column order of &lt;data_sources&gt;.</td>
</tr>
</tbody>
</table>
### Match type

<table>
<thead>
<tr>
<th>Match type</th>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>T(A)</td>
<td>because it does not contain the exact same columns and column order of <code>&lt;data_sources&gt;</code>.</td>
<td></td>
</tr>
<tr>
<td>T(X,A,B,C)</td>
<td>because T(X) is not a column in <code>&lt;data_sources&gt;</code>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CASCADE</th>
<th>T(A,C)</th>
<th>T(X) - because it does not contain any columns that match the columns in <code>&lt;data_sources&gt;</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A,B,C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A,X)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(C,B,A,X)</td>
<td></td>
</tr>
</tbody>
</table>

### match_properties

Specifies properties to use for matching when selecting data statistics.

```xml
<match_properties> ::= <match_property>[...]
<match_property> ::= TYPE <data_statistics_type> | REFRESH TYPE <refresh_type_filter>
```

If TYPE is not specified, then all data statistics objects of any type on the specified data sources are refreshed (ALL). For descriptions of the supported data statistics types see the `CREATE STATISTICS Statement` topic.

### data_statistics_type

Specifies the type of data statistics objects to match when selecting the data statistics.

```xml
<data_statistics_type> ::= TYPE <type_name>
<type_name> ::= HISTOGRAM | SIMPLE | TOPK | SKETCH | SAMPLE | RECORD COUNT | ALL
```

### refresh_type_filter

Specifies the refresh strategy to match in the data statistics objects when selecting the data statistics to refresh. ALL is the default.

```xml
<refresh_type_filter> ::= AUTO | MANUAL | ALL
```
Description

You can specify multiple filtering clauses to refine the list of data statistics objects to drop.

Refer to the CREATE STATISTICS statement for complete descriptions of the data statistic types (<data_statistics_type> and <refresh_type>).

Permissions

One of the following is true:

• You own the object you are dropping statistics from.
• You have the ALTER privilege on the object you are dropping statistics from.
• For linked database, you have the LINKED DATABASE object level privilege on the remote source, regardless of who owns the remote source.

Example

Drop the HISTOGRAM on column T.X, where T is a virtual table:

```
DROP STATISTICS ON MYSHEMA.T(X) TYPE HISTOGRAM;
```

Drop a data statistics object with the name HISTOGRAM_T_X:

```
DROP STATISTICS MYSHEMA.HISTOGRAM_T_X;
```

Drop any data statistic object with <data_sources> defined as T(A,B):

```
DROP STATISTICS ON MYSHEMA.T(A,B) EXACT;
```

Drop any data statistic object that has table columns T.A, T.B, or T.C listed in <data_sources>m. For example, SKETCH and SAMPLE defined on T(A, B).

```
DROP STATISTICS ON MYSHEMA.T(A,B,C) CASCADE TYPE ALL;
```

Drop the HISTOGRAMS on table columns T.A, T.B, and T.C:

```
DROP STATISTICS ON MYSHEMA.T(A,B,C) TYPE HISTOGRAM;
```

Drop the record count on table T:

```
DROP STATISTICS ON MYSHEMA.T TYPE RECORD COUNT;
```
4.10.1.113 DROP STRUCTURED PRIVILEGE Statement (Access Control)

Drops a structured (analytic) privilege.

Syntax

```
DROP STRUCTURED PRIVILEGE <privilege_name>
```

Syntax Elements

- `privilege_name`
  
  Specifies the name of the analytic privilege to drop.

Permissions

You must have the STRUCTUREDPRIVILEGE ADMIN privilege to execute this statement.

Description

Drops an analytic privilege. Analytic privileges provide fine-grained control over which data a user can see within a view. Analytic privileges are sometimes referred to as structured privileges.
4.10.1.114 DROP SYNONYM Statement (Data Definition)

Removes a synonym.

Syntax

```
DROP [PUBLIC] SYNONYM <synonym_name> [<drop_option>]
```

**Syntax Elements**

**PUBLIC**
Allows the removal of a public synonym.

**SYNONYM synonym_name**
Drops the specified synonym, with the optional schema name.

```
<synonym_name> ::= [<schema_name>.]<identifier>
<schema_name> ::= <unicode_name>
```

**drop_option**
Specifies the drop option to use.

```
<drop_option> ::= CASCADE | RESTRICT
```

CASCADE drops the synonym and dependent objects. RESTRICT drops the synonym only when dependent objects do not exist. If this drop option is used and a dependent object exists, then an error is thrown.

If `drop_option` is not specified, then a non-cascaded drop is performed which only drops the specified synonym. Dependent objects of the synonym are invalidated but not dropped.

The invalidated object can be re-validated when an object that has same schema and object name is created.
Description

The DROP SYNONYM statement removes a synonym.

Permissions

Only database users with DROP object privilege on the remote source can drop a synonym from a remote source.

Example

Create a table a, then a synonym, a_synonym, and a public synonym, pa_synonym, for the table.

```
CREATE ROW TABLE a (c INT);
CREATE SYNONYM a_synonym FOR a;
CREATE PUBLIC SYNONYM pa_synonym FOR a;
```

Drop both the synonym a_synonym and public synonym pa_synonym using a non-cascaded drop in both cases.

```
DROP SYNONYM a_synonym;
DROP PUBLIC SYNONYM pa_synonym;
```

4.10.1.115 DROP TABLE Statement (Data Definition)

Removes a physical or virtual table from the database.

Syntax

```
DROP TABLE <table_name> [<drop_option>] [ WITH REMOTE ]
```

Syntax Elements

table_name

Drops the specified table or collection table, with the optional schema name.

```
<table_name> ::= [ [ <database_name>.]<schema_name>.]<identifier>
```
<schema_name> ::= <unicode_name>

drop_option

Specifies the drop option to use.

<drop_option> ::= CASCADE | RESTRICT

CASCADE drops the table and dependent objects. RESTRICT drops the table only when dependent objects do not exist. If this drop option is used and a dependent object exists, then an error is thrown.

If <drop_option> is not specified, then a non-cascaded drop is performed which only drops the specified table. Dependent objects of the table are invalidated but not dropped.

The invalidated objects can be re-validated when an object that has same schema and object name is created.

WITH REMOTE Drops the virtual table from the local source and the corresponding table on the remote source. This option is only supported for virtual tables. The WITH REMOTE option is supported for remote sources using the following SDA adapters: hanaodbc, iqodbc, aseodbc, tdodbc, voraodbc, odbc (Oracle, Microsoft SQL Server, IBM Netezza, IBM DB2, BigQuery, PostgreSQL). It is not supported with an SDI adapter.

Description

The DROP TABLE statement removes a physical or virtual table from the database.

Permissions

This statement requires the DROP object privilege on the table.

Use of the WITH REMOTE option also requires the REMOTE TABLE ADMIN object privilege.

When using the WITH REMOTE option, if the credential type of the remote source is PASSWORD, then only the owner of the remote sources who set the technical user credentials has the authorization to use the option. For all other credential types, authorization is a function of the privileges of the user on the remote source.

Example

Drop table A and all its dependent objects.

```sql
DROP TABLE A;
```
4.10.1.116 DROP TRIGGER Statement (Data Definition)

Deletes a trigger.

Syntax

```
DROP TRIGGER <trigger_name> [ <drop_option> ] [ ONLINE ]
```

Syntax Elements

**trigger_name**

Drops the specified trigger, with the optional schema name.

```
<trigger_name> ::= [ <schema_name>. ] <identifier>
<schema_name> ::= <unicode_name>
```

**drop_option**

Specifies the drop option to use.

```
<drop_option> ::= CASCADE | RESTRICT
```

CASCADE drops the trigger and dependent objects. RESTRICT drops the trigger only when dependent objects do not exist. If this drop option is used and a dependent object exists, then an error is thrown.

If `drop_option` is not specified, then a non-cascaded drop is performed, which only drops the specified trigger. Dependent objects of the trigger are invalidated but not dropped.

**ONLINE**

Applies an intentional exclusive lock (instead of an exclusive lock) before performing the operation. An intentional exclusive lock differs from the default lock (exclusive) because it allows concurrent DML and DDL related to the object to proceed, and will retry the operation until it can proceed.

ONLINE applies an intentional exclusive lock only if the auto commit mode for DDL statements is set to on. If the auto commit mode for DDL statement is not set to on, then an exclusive lock is still applied.
**Description**

Invalidated objects can be re-validated when an object that has same schema and object name is created.

**Permissions**

Only database users with the TRIGGER privilege for the table on which the trigger was defined are allowed to drop a trigger for that table.

**Example**

Create two tables, `target` and `sample`, and a trigger called `test`.

```sql
CREATE ROW TABLE target ( A INT);
CREATE ROW TABLE sample ( A INT);
CREATE TRIGGER test
AFTER UPDATE ON target
BEGIN
  INSERT INTO sample VALUES(3);
END;
```

Drop the trigger called `test`.

```sql
DROP TRIGGER test;
```

Alternatively, you can drop the trigger and specify ONLINE so that the operation proceeds only when concurrent DML or DDL finishes.

```sql
DROP TRIGGER test ONLINE;
```

**4.10.1.117 DROP TYPE Statement (Procedural)**

Removes a user-defined table type.

**Syntax**

```
DROP TYPE <type_name> [<drop_option>]
```
Syntax Elements

type_name
Specifies the table type to be dropped with optional schema name.

\[<type_name> ::= [<schema_name>.]<identifier>\]
\[<schema_name> ::= <unicode_name>\]

drop_option
Specifies the desired drop behavior.

\[<drop_option> ::= CASCADE | RESTRICT\]

The default behavior is to drop only the specified table type, regardless of dependencies. When RESTRICT is specified, an error is returned if there are any objects (currently only tables are supported) that are dependent on the type. When CASCADE is specified, both the specified type and any dependent objects are dropped. For example, assume there is a type T being used by a database object O:

- DROP TYPE T - Only type T is dropped; object O becomes invalid.
- DROP TYPE T RESTRICT - The drop operation fails and an error is returned (because O is dependent on type T).
- DROP TYPE T CASCADE - Both type T and object O are dropped.

Description

The DROP TYPE statement removes a user-defined table type.

To find invalid objects in the database, look at the IS_VALID columns in the relevant system view. For example, you can look at the IS_VALID column of the TABLES system view to determine which tables are valid.

Invalidated objects can be revalidated when an object that has same schema and object name is created.

Example

Create a table type called my_type.

```
CREATE TYPE "my_type" AS TABLE ( "column_a" DOUBLE );
```

Drop the my_type table type.

```
DROP TYPE "my_type";
```
4.10.1.118  DROP USER Statement (Access Control)

Deletes a database user.

Syntax

```
DROP USER <user_name> [drop_option]
```

Syntax Elements

- **user_name**
  
  Drops the specified user.
  
  ```
  <user_name> ::= <unicode_name>
  ```
  
  `<user_name>` must specify an existing database user.

- **drop_option**
  
  Specifies the drop option to use.
  
  ```
  <drop_option> ::= CASCADE | RESTRICT
  ```

  **CASCADE** drops the user and performs the following actions:
  
  - Drops the user's home schema and other schemas belonging to the user, together with all objects stored in them regardless of which user created them.
  - Drops all objects owned by the user, even if they are part of another schema.
  - Drops objects that are dependent on deleted objects and that are owned by the user being dropped.
  - Invalidates objects that are dependent on deleted objects and that are owned by a different user in a schema not belonging to the user being dropped.
  - Drops public synonyms that are owned by the user being dropped.
  - Revokes privileges on deleted objects.
  - Revokes privileges granted by the deleted user. Revoked privileges may cause further revokes if the privileges have been granted further.

  **RESTRICT** drops the user only if they don't own any schemas or objects other than their home schema. If RESTRICT is specified while the user owns any objects, then an error is returned.

  When `<drop_option>` is not specified, RESTRICT is used by default.

Description

Users created by the deleted user and roles created by them are not deleted.
Audit policies created by the deleted user are not deleted. It is possible to delete a user even if an open session for this user exists. The deleted user is deleted in the following views: USERS, USER_PARAMETERS, INVALID_CONNECT_ATTEMPTS.

The deletion of objects influence all of the system views describing objects: TABLES, VIEWS, PROCEDURES, and so on.

The deletion of objects influence the view describing privileges like GRANTED_PRIVILEGES and all of the monitoring views like M_RS_TABLES, M_TABLE_LOCATIONS, and so on.

**Permissions**

Only database users with the USER ADMIN system privilege can drop a database user. Each user with this privilege is allowed to drop any user. Only those users which are delivered with the SAP HANA database cannot be dropped: SYS, SYSTEM, _SYS_REPO, and _SYS_STATISTICS.

**System Views**

**Example**

Create a new user called new_user.

```sql
CREATE USER new_user PASSWORD Password1;
```

Drop the user new_user you created in the previous step. As the CASCADE option is used, the user is dropped together with all of the objects belonging to that user.

```sql
DROP USER new_user CASCADE;
```

**Related Information**

- USERS System View [page 1698]
- USER_PARAMETERS System View [page 1702]
- INVALID_CONNECT_ATTEMPTS System View [page 1588]
- M_RS_TABLES System View [page 2089]
- M_TABLE_LOCATIONS System View [page 2204]
4.10.1.119 DROP USERGROUP Statement (Access Control)

Removes a user group from the database.

Syntax

```
DROP USERGROUP <usergroup_name>
```

Syntax Elements

- `usergroup_name`
  - The name of the user group.

Description

You must remove all users from the user group before you can drop it. An error is returned if you attempt to drop a user group that still has users.

Permissions

You must have the USER ADMIN privilege to drop a user group.

Example

The following statement drops a (fictitious) user group named `MyUserGroup`:

```
DROP USERGROUP MyUserGroup;
```

Related Information

- ALTER USER Statement (Access Control) [page 654]
- CREATE USERGROUP Statement (Access Control) [page 902]
- ALTER USERGROUP Statement (Access Control) [page 665]
4.10.1.120 DROP VIEW Statement (Data Definition)

Removes a view from the database.

Syntax

DROP VIEW <view_name> [<drop_option>]

Syntax Elements

**view_name**

Drops the specified view, with the optional schema name.

```
<view_name> ::= [ <schema_name>. ]<identifier>
<schema_name> ::= <unicode_name>
```

**drop_option**

Specifies the drop option to use.

```
<drop_option> ::= CASCADE | RESTRICT
```

CASCADE drops the view and dependent objects. RESTRICT drops the view only when dependent objects do not exist. If this drop option is used and a dependent object exists, then an error is thrown.

If `<drop_option>` is not specified, then a non-cascaded drop is performed which only drops the specified view. Dependent objects of the view are invalidated but not dropped.

The invalidated objects can be re-validated when an object that has same schema and object name is created.

Description

The DROP VIEW statement removes a view from the database.

Dropping a view also drops any expression macros defined for the view. The list of all expression macros for a view can be obtained by querying the VIEW_EXPRESSION_MACROS system view.
Example

Create table t, and then a view, v, that selects all records from table t.

```
CREATE ROW TABLE t (a INT);
CREATE VIEW v AS SELECT * FROM t;
```

Drop view v from the database.

```
DROP VIEW v;
```

Related Information

VIEW_EXPRESSION_MACROS System View [page 1708]

4.10.1.121 DROP WORKLOAD CLASS Statement (Workload Management)

Removes workload classes.

Syntax

```
DROP WORKLOAD CLASS <workload_class_name>
```

Syntax Elements

```
workload_class_name

Removes the specified workload class.

<workload_class_name> ::= <identifier>
```

Description

Removes workload classes.
Permissions

The use of this statement requires the WORKLOAD_ADMIN privilege.

Examples

Drop the workload class “MyWorkloadClass”.

```
DROP WORKLOAD CLASS "MyWorkloadClass";
```

Related Information

WORKLOAD_CLASSES System View [page 1721]

4.10.1.122  DROP WORKLOAD MAPPING Statement
(Workload Management)

Drops a workload mapping.

Syntax

```
DROP WORKLOAD MAPPING <mapping_name>
```

Syntax Elements

- **mapping_name**
  
  Specifies the workload mapping to drop.

```
<mapping_name> ::= <identifier>
```

Description

Drops a workload mapping.
Permissions

The use of this statement requires the WORKLOAD_ADMIN privilege.

Examples

Remove the workload mapping "MyWorkloadMapping".

```
DROP WORKLOAD MAPPING "MyWorkloadMapping";
```

Related Information

WORKLOAD_MAPPINGS System View [page 1723]

4.10.1.123 DROP X509 PROVIDER (Access Control)

Drops an X.509 provider in the SAP HANA database.

Syntax

```
DROP X509 PROVIDER <x509_provider_name> [ CASCADE ]
```

Syntax Elements

- **x509_provider_name**
  
  Specifies which X.509 provider to drop.

  ```
  <x509_provider_name> ::= <simple_identifier>
  ```

  - **<x509_provider_name>** must be an existing X.509 provider.

- **CASCADE**
  
  Required only when the X.509 provider is referenced by a PSE’s purpose object. The provider is removed from any PSE referencing it along with any user mappings for users authenticating with the provider. For PSEs where it is the only provider assigned, the purpose of that PSE is also removed.
Permissions

Only database users with the USER ADMIN system privilege can drop an X.509 provider.

Examples

Drops the X.509 provider, my_x509_provider1.

```
DROP X509 PROVIDER my_x509_provider1;
```

Related Information

- X.509 Certificate-Based User Authentication
- CREATE X509 PROVIDER (Access Control) [page 937]
- ALTER X509 PROVIDER (Access Control) [page 684]
- X509_PROVIDERS System View [page 1725]
- X509_PROVIDER_RULES System View [page 1725]
- X509_USER_MAPPINGS System View [page 1726]

4.10.1.124 EXPLAIN PLAN Statement (Data Manipulation)

Evaluates the execution plan that the database follows when executing an SQL statement.

Syntax

```
EXPLAIN PLAN [ SET STATEMENT_NAME = <statement_name> ] FOR <explain_plan_entry>
```

Syntax Elements

statement_name

Specifies the name of a specific execution plan in the output table for a given SQL statement.

```
<statement_name> ::= <string_literal>
```

explain_plan_entry
Specifies the entry to explain.

\[
<\text{explain\_plan\_entry}> ::= \\
    <\text{subquery}> \\
    | \text{SQL PLAN CACHE ENTRY <plan\_id>} \\
<\text{plan\_id}> ::= \text{<integer\_literal>}
\]

The \( <\text{plan\_id}> \) specifies the identifier of an entry to be explained, which is already in the SQL plan cache. Refer to the \text{M\_SQL\_PLAN\_CACHE} monitoring view to find the \( <\text{plan\_id}> \) for the desired cache entry.

\text{subquery}

Specifies the subquery to explain the plan for. For more information about subqueries, see the \text{SELECT} statement.

**Permissions**

The \text{EXPLAIN PLAN [SET STATEMENT\_NAME = <statement\_name>] FOR SQL PLAN CACHE ENTRY <plan\_id>} command can only be run by users with the \text{OPTIMIZER\_ADMIN} privilege.

**Description**

Using this command, a user can see the execution plan of a subquery, or that of an entry already in the SQL plan cache. The result of the evaluation is stored in the \text{EXPLAIN\_PLAN\_TABLE} view for examination.

The SQL \( <\text{subquery}> \) must be one of the following type of statements: \text{INSERT}, \text{UPDATE}, \text{DELETE}, \text{REPLACE}, \text{UPsert}, \text{MERGE INTO} and \text{SELECT}. All other type of statements cannot be used with \text{EXPLAIN PLAN}. The same rule is applied to the plan cache entry. That is, the cache entry a user specified using \( <\text{plan\_id}> \) should be a one of the statements listed above.

**Example**

In the \text{OPERATOR\_NAME} of the \text{EXPLAIN\_PLAN\_TABLE} system view:

- \text{COLUMN SEARCH} denotes the starting position of column engine operators.
- \text{ROW SEARCH} denotes the starting position of row engine operators.

In the example below, the intermediate result produced by a \text{COLUMN SEARCH (ID 10)} is consumed by a \text{ROW SEARCH (ID 7)}. The intermediate result produced by the \text{ROW SEARCH (ID 7)} is consumed by another \text{COLUMN SEARCH (ID 1)}.

The operators below the lowest \text{COLUMN SEARCH (ID 10)} explain how the \text{COLUMN SEARCH (ID 10)} is executed.

The operators between the \text{ROW SEARCH (ID 7)} and the \text{COLUMN SEARCH (ID 10)} explain how the \text{ROW SEARCH (ID 7)} processes the intermediate result produced by the \text{COLUMN SEARCH (ID 10)}. 
The operators between the top COLUMN SEARCH (ID 1) and the ROW SEARCH (ID 7) explain how the top COLUMN SEARCH (ID 1) processes the intermediate result produced by the ROW SEARCH (ID 7).

<table>
<thead>
<tr>
<th>OPERATOR_NAME</th>
<th>OPERATOR_ID</th>
<th>PARENT_OPERATOR_ID</th>
<th>LEVEL</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMN SEARCH</td>
<td>1</td>
<td>NULL</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LIMIT</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>GROUP BY</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>JOIN</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>COLUMN TABLE</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>ROW SEARCH</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>BTREE INDEX JOIN</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>BTREE INDEX JOIN</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>COLUMN SEARCH</td>
<td>10</td>
<td>9&lt;</td>
<td>&gt;9</td>
<td>1</td>
</tr>
<tr>
<td>FILTER</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>COLUMN TABLE</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>1</td>
</tr>
</tbody>
</table>

**SQL plan explanation**

Here is an example query plan explanation. In the example, all tables are located on row store.

```sql
DELETE FROM explain_plan_table WHERE statement_name = 'TPC-H Q10';
EXPLAIN PLAN SET STATEMENT_NAME = 'TPC-H Q10' FOR
SELECT TOP 20
  c_custkey,
  c_name,
  SUM(l_extendedprice * (1 - l_discount)) AS revenue,
  c_acctbal,
  n_name,
  c_address,
  c_phone,
  c_comment
FROM
  customer,
  orders,
  lineitem,
  nation
WHERE
  c_custkey = o_custkey
AND l_orderkey = o_orderkey
AND o_orderdate >= '1993-10-01'
AND o_orderdate < ADD_MONTHS('1993-10-01',3)
AND l_returnflag = 'R'
AND c_nationkey = n_nationkey
GROUP BY
  c_custkey,
```
SELECT operator_name, operator_details, table_name
FROM explain_plan_table
WHERE statement_name = 'TPC-H Q10';

DELETE FROM explain_plan_table WHERE statement_name = 'TPC-H Q10';
EXPLAIN PLAN SET STATEMENT_NAME = 'TPC-H Q10' FOR SQL PLAN CACHE ENTRY 2050002;
SELECT operator_name, operator_details, table_name
FROM explain_plan_table
WHERE statement_name = 'TPC-H Q10';

The two examples give the same following result table.
From this plan result we can obtain the following facts about this query:

- TABLE SCAN is executed on ORDERS with the FILTER CONDITION.
- BTREE INDEX JOIN is executed with the B-tree index of CUSTOMER and the result of the TABLE SCAN below.
• BTREE INDEX JOIN is executed with the B-tree index of NATION and the result of the BTREE INDEX JOIN below.
• CPBTREE INDEX JOIN is executed with the CPB-tree index of LINEITEM and the result of the BTREE INDEX JOIN below.
• GROUP BY is executed with the result of the CPBTREE INDEX JOIN below, with 4 threads.
• MERGE AGGREGATION is executed with the result of the GROUP BY below.

4.10.1.125 EXPORT Statement (Data Import Export)

Exports catalog objects.

Syntax

```
EXPORT <export_object_name_list>
[ WHERE <condition> | HAVING <condition_list> ]
[ AS <export_format> ]
INTO { <path> [ <archive_file_name> ]
| <storage_path> }
[ WITH <export_option_list> ]
[ <query_export_specification> ]
```

Syntax Elements

**export_object_name_list**

Specifies a list of objects to export. The ALL and * options export all objects from a specified schema.

```
<export_object_name_list> ::= <export_object_name>[, <export_object_name>[,...]]
| CLIENTSIDE ENCRYPTION COLUMN KEY <encryption_column_key_name>[, <encryption_column_key_name>[,...]]
| { ALL | * }
<export_object_name> ::= [schema_name].{<identifier> | "*" }
<encryption_column_key_name> ::= [schema_name].{<identifier> }
```

**export_object_name**

Specifies objects in a schema for export. Specify `<schema_name>"*"` to select all objects within the specified schema for export. Specify `ALL` (without `<schema_name>`) to select all objects from all schemas in the system for export. `<schema_name>"*"` is not supported with the HAVING clause.

**encryption_column_key_name**

Specifies the name of a column encryption key (CEK).

**WHERE condition**
Exports a subset of table data.

```
WHERE <condition> ::=  
|<condition> OR <condition>  
|<condition> AND <condition>  
| NOT <condition>  
| ( <condition> )  
| <predicate>  
<predicate> ::=  
|<comparison_predicate>  
|<range_predicate>  
|<in_predicate>  
|<exist_predicate>  
|<like_predicate>  
|<null_predicate>  

<comparison_predicate> ::=  
|<expression> { = | != | <> | > | < | >= | <= } [ ANY | SOME | ALL ] ( { 
|<expression> | <subquery> } )  

(range_predicate) ::=  
|<expression> [ NOT ] BETWEEN <expression> AND <expression>  

(in_predicate) ::=  
|<expression> [ NOT ] IN ( { <expression_list> | <subquery> } )  

(exist_predicate) ::=  
|NOT] EXISTS ( <subquery> )  

(like_predicate) ::=  
|<expression> [ NOT ] LIKE <expression> [ ESCAPE <expression> ]  

(null_predicate) ::=  
|<expression> IS [ NOT ] NULL  

expression_list ::= <expression> [, <expression> [,...] ]
```

The WHERE clause follows, and is associated with, a single table in the EXPORT statement. If at least one table has a WHERE clause specified, then the following restrictions are enforced:

- Only the CSV format is supported for export
- The following options are not allowed: CATALOG ONLY, STATISTICS ONLY, and NO DEPENDENCIES

The WHERE clause cannot be used in conjunction with ALL or `*`.

**HAVING**

Exports a subset of data from a specified object.

```
HAVING <condition_list>
```

The HAVING clause cannot be used in conjunction with `<schema>.<identifier>` or `<schema>.*`.

For the types of conditions you can specify, see `<condition_list>`.

**condition_list**

Specifies the conditions defined for the HAVING and DEPENDENCIES clauses.

```
<condition_list> ::=  
|<condition> OR <condition>  
|<condition> AND <condition>  
| NOT <condition>  
| ( <condition> )  
| <predicate>  
```
**AS export_format**

Specifies the format to export the data to. The default value is BINARY.

```plaintext
<export_format> ::= CSV | BINARY [ <binary_type> ]
<binary_type> ::= DATA | RAW
```

**BINARY** Table data is exported in an internal binary format. Exporting in BINARY RAW format is orders of magnitude faster than exporting the same table in CSV format. Only column non-temporary tables can be exported in binary format. Row tables are always exported in CSV format regardless of the format specified. If `<binary_type>` is not specified, RAW (default) is used. Binary DATA is compatible between SAP HANA Platform (on-premise) and SAP HANA Cloud.

**RAW**

The raw binary format of the data as it is stored in the physical storage layer of the database. This format is specific to the product implementation and is only compatible with the SAP HANA product it was generated from.

**IMPORTANT:** This format is not compatible between SAP HANA Platform or SAP HANA Service (based on SAP HANA Platform) and SAP HANA database. It cannot be used for transferring data between these products.

**DATA** A binary representation of the data values stored in the database. This format is not based on the physical storage layer. It is compatible between SAP HANA Platform and SAP HANA database and can be used to transfer data between the different SAP HANA products.

**CSV** Table data is exported in CSV format. With this export format the exported data can be scrambled. Both column and row tables can be exported in CSV format.

**INTO path [ archive_file_name ]**

Specifies the location where export files are placed.

```plaintext
<path> [ <archive_file_name> ] ::= <string_literal>
```

Include the file extension .tar.gz or .tgz on `<archive_file_name>` to export the data to an archive file. Each file added to the archive file must not exceed 8 GB in size or it will become corrupt. There is no...
size restriction on the archive file itself. See SAP Note 2907201 - Importing Data From Compressed Format Throws Error: “Archive is incomplete”.

For security reasons, the path must not contain symbolic links and must not point inside the database instance folder, except its backup and work subfolders. Examples of valid values include:

' /tmp'
' /usr/sap/HDB/HDB00/backup'
' /usr/sap/HDB/HDB00/work/myexport.tgz'

In distributed systems, `<path>` must be a shared disk; otherwise, the operation may fail or have unexpected results.

**storage_path**

Specifies the cloud storage location for the export.

<storage_path> ::=  
{ <azure_path>  
| <amazon_path>  
| <google_path>  
| <hdlfs_path>  }

**azure_path**

Specifies the location for the Azure export file.

<azure_path> ::=  
'azure://[<azure_credentials>@]<azure_container_name>/<azure_object_id>

**azure_credentials** Required when not using the WITH CREDENTIAL parameter.

<azure_credentials> ::=  
<azure_storage_account_name>:<SAS-token>

**azure_container_name** Specifies the name of the Azure container to access storage.

**azure_object_id** Specifies a path to the object within the Azure storage.

**amazon_path**

Specifies the location for the Amazon (AWS) export file.

<amazon_path> ::=  
's3-<amazon_region>://[<amazon_credentials>@]<amazon_bucket_name>/<amazon_object_id>

**amazon_region** Specifies the geographical region the bucket is located in. Refer to Regions and Availability Zones.

**amazon_credentials** Not supported with the WITH CREDENTIAL parameter. Specifies the credential key pair for API access from the AWS IAM Management Console. This is not the AWS account.

<amazon_credentials> ::=  
'<amazon_access_key>:<amazon_secret_key>

**amazon_bucket_name** Specifies the name assigned to the Amazon storage bucket when it was created.

**amazon_object_id** Specifies a path to the object within the named bucket.

**google_path**
Specifies the location for the Google Cloud storage export file.

```plaintext
<google_path> ::= 'gs://[<google_credentials>@]<google_bucket_name>/<google_object_id>
```

**google_credentials** Not supported with the WITH CREDENTIAL parameter. Specifies the credential key pair for access from the Google IAM Management Console. This is not the Google Cloud account.

```plaintext
<google_credentials> ::= <google_service_account>:<google_private_key>
```

**google_bucket_name** Specifies the name assigned to the Google Cloud storage bucket when it was created.

**google_object_id** Specifies a path to the object within the named bucket.

**hdlfs_path**

Specifies the location for the SAP HANA Cloud, Data Lake Files (HDLFS).

```plaintext
<hdlfs_path> ::= hdlfs://<hdlfs_endpoint>/<hdlfs_object_id>
```

**hdlfs_endpoint** Specifies the server address given from data lake Files. It consists of the data lake Files container id and the landscape.

**hdlfs_object_id** Specifies a path or file name within the data lake Files container.

**WITH export_option_list**

Specifies a list of export options.

```plaintext
WITH <export_option_list>
<export_option_list> ::= <export_option>[, <export_option>[,...] ]
<export_option> ::= REPLACE | CATALOG ONLY | NO DEPENDENCIES | DEPENDENCIES <condition_list> | SCRAMBLE [BY <password>] | STRIP | THREADS <number_of_threads> [ [ SYSTEM | ALL ] STATISTICS [ COLUMNS <column_list> ] [ TYPE <dstat_type> ] [ ONLY ] ] | NO STATISTICS | TRACE | PERSISTENT MEMORY | NO REMOTE DATA | CREDENTIAL '<purpose_def>'
```

**REPLACE**

Defines the behavior if the export data already exists in the specified directory. If REPLACE is not specified, then an error is returned if previously exported data exists in the specified export directory.

**CATALOG ONLY**

Specifies to export only the database catalog.

This option interacts with the STATISTICS ONLY and NO STATISTICS options.

**NO DEPENDENCIES**

Specifies to not export the underlying dependencies of an export object.
**DEPENDENCIES**

Specifies the underlying dependencies of an object to export.

```
DEPENDENCIES <condition_list>
```

For the types of conditions you can specify, see `<condition_list>`.

DEPENDENCIES is not supported with the WHERE clause.

**SCRAMBLE [BY scramble_seed]**

Obfuscates CSV format exported data.

```
SCRAMBLE [BY <scramble_seed>]  
<scramble_seed> ::= <string_literal>
```

Provides an export of the structure of your database with the string data scrambled so that it cannot easily be interpreted. When the optional `<scramble_seed>` is not specified, a default scramble seed is used. Make the `<scramble_seed>` at least 8 to 10 characters long.

Only CSV format character string data is scrambled by this parameter.

**STRIP**

Strips attributes that can be reconstructed at import time from the export data. This option can reduce the size of the export file in some cases.

**THREADS number_of_threads**

Specifies the number of process threads to be used for concurrent export processing.

```
THREADS <number_of_threads>  
<number_of_threads> ::= <unsigned_integer>
```

The THREADS parameter specifies how many objects are exported in parallel; the default is 1. Increasing the number of threads reduces export time, but can also negatively affect database system performance. Consider the following items when you use this parameter:

- When exporting a single table. THREADS has no effect.
- When exporting a view or procedure, two or more threads should be used, up to the number of dependent objects.
- When exporting a whole schema, consider using more than 10 threads, with a maximum being the number of CPU cores in the system.
- When exporting a whole BW/ERP system database with tens of thousands of tables by using the ALL keyword, a large number of threads can be used (up to 256).

**PERSISTENT MEMORY**

Exports the persistent memory data as files for columns that are actively using persistent memory. If this option is specified and persistent memory is not configured for the database, then the EXPORT statement fails.

```
[ SYSTEM | ALL ] STATISTICS [ COLUMNS column_list ] [ TYPE dstat_type ] [ ONLY ]
```

User-defined data statistics objects are exported by default, along with the objects they reference in `<export_object_name_list>` (for example, tables).
SYSTEM statistics are only available on column and row tables.

If you specify SYSTEM, then system statistics that were automatically generated for the specified data sources are exported. If no system statistics are available for a data source, then no data statistics objects are exported. If you specify ALL, then both system- and user-defined data statistics are exported, where available. If both system- and user-defined statistics are available on a data source, then only the user-defined statistics are exported. If neither SYSTEM nor ALL is specified, then only user-defined data statistics are exported. This is the default behavior in the absence of the STATISTICS clause.

If <column_list> is specified, then any data statistics objects that reference all or some of the specified columns, but none of the other columns are exported. When you export SYSTEM data statistics, <column_list> is required when the statistics type SAMPLE is specified. If <column_list> is not specified and SYSTEM statistics are requested, then TOPK and SIMPLE data statistics for each column of the specified data object are exported, if available. If <column_list> is specified, then only a single data object can be specified. If <column_list> is specified and SYSTEM statistics are requested, then data statistics of type SIMPLE, TOPK, and SAMPLE are exported. SIMPLE and TOPK are generated for every eligible column on the list, while a single SAMPLE covering all eligible columns from the specified list is generated.

<dstat_type> filters exported data statistics to only the specified type. For SYSTEM data statistics, supported types are SIMPLE, TOPK, and SAMPLE.

If you specify STATISTICS ONLY without CATALOG ONLY, then both the data and metadata for user-defined data statistics objects are exported, as well as the metadata for the objects in <export_object_name_list>.

If you specify STATISTICS ONLY and CATALOG ONLY, then only the metadata for the data statistics objects and the objects in <export_object_name_list> are exported; no data is exported.

NO STATISTICS

Excludes both system- and user-defined data statistics objects from the export. Metadata and data for the objects in <export_object_name_list> are still exported, and they are impacted as normal by whether CATALOG ONLY is specified.

TRACE export_trace_file_name

Writes execution information to the specified trace file when exporting data.

The info level trace file contains detailed error descriptions for troubleshooting. The file is saved to the current trace directory.

The contents of the trace file can be found in the M_TRACEFILE_CONTENTS system view. It contains information on:

- The export list
- Dependencies to the export list
- Feasibility checks on the export
- Metadata exported
The format of the exported data (binary or CSV)
The single archive for a single archive export
The results table from the export

To delete the trace file, use the ALTER SYSTEM REMOVE TRACES statement.

**NO REMOTE DATA**
Excludes data in remote records from the export. When the EXPORT statement references virtual tables, remote table data is included in the export if you don’t specify CATALOG ONLY. Including remote table records can significantly slow the export process.

**CREDENTIAL purpose_def**
Specifies the name of the credential defined in the CREATE CREDENTIAL statement. Since the credentials are defined within the credential, they no longer appear as plain text as part of export statements. The WITH CREDENTIAL clause cannot be specified when `<cloud_path>` contains credentials. The WITH CREDENTIAL clause is required for exports to SAP HANA Cloud, Data Lake Files, but is optional for all other cloud storage locations.

**query_export_specification**
Specifies a query to use for the export.

```sql
<query_export_specification> ::=    [ ON <sqlscript_location_list> ] FOR <procedure_call_statement>
```

For information on query exports, see the *SAP HANA Cloud, SAP HANA Database SQLScript Reference*.

**Permissions**
Requires the following:

- EXPORT system privilege
- Privilege to access the object to export:
  - SELECT object privilege on the TABLE, VIEW, SEQUENCE, or GRAPH WORKSPACE
  - EXECUTE privilege on PROCEDURE or FUNCTION
  - CREATE TEMPORARY TABLE privilege on the current schema

To export column encryption keys (CEKs), you must also have the USAGE privilege on the CEK.

The EXPORT statement is not audited.

**Description**
The EXPORT command exports tables, views, column views, synonyms, data statistics objects, sequences, procedures, or column encryption keys in the specified format as BINARY or CSV.

The EXPORT command does not support parameterized views.

System-versioned tables and their associated history table must be exported separately and must be exported by using AS BINARY.
Data stored in "no logging" tables can only be exported in CSV format. For global temporary tables, only the table catalog can be exported. The export of local temporary tables is not supported.

You can only import exported binary files that were generated on an SAP HANA system with the same endianness.

You can cancel an export session by using the ALTER SYSTEM CANCEL WORK IN SESSION statement and specifying the connection ID from the M_EXPORT_BINARY_STATUS view.

Detailed results of the last successful execution of the EXPORT statement are stored in the session-local temporary table `<current_schema>.#EXPORT_RESULT`. If EXPORT has not been executed in the current session, then selecting from `<current_schema>.#EXPORT_RESULT` returns an error. If `<current_schema>` of the session is invalid, then EXPORT throws an exception as it cannot create its result table in `<current_schema>.#EXPORT_RESULT`.

The user exporting tables with encrypted columns must export all relevant column encryption keys (CEKs) used by this table. Exporting a CEK exports all CEK copies of the CEK. The export of a CEK copy also includes the public key of the relevant client key pairs.

### Note

When using CSV format to export/import tables that have columns of type `GENERATED ALWAYS AS <expression>`, verify that the imported records match what was exported. If there are anomalies:

- For column tables, switch to exporting/importing as `BINARY`.
- For row tables, change the table to be a column table and export/import as `BINARY`.

For more information on security on exported files, refer to the [SAP HANA Security Checklists and Recommendations](#) guide.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>The database of the exported object</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>The schema of the exported object</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>The name of the exported object</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>The type of the exported object (TABLE, VIEW, and so on)</td>
</tr>
<tr>
<td>LOCATION</td>
<td>VARCHAR(75)</td>
<td>The location <code>&lt;host&gt;:&lt;port&gt;</code> where the object was exported</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>The export status (done, skipped, failed)</td>
</tr>
<tr>
<td>ERROR</td>
<td>VARCHAR(512)</td>
<td>The error text if there is an export failure</td>
</tr>
</tbody>
</table>
Examples

This example exports a client-side encryption key:

```
EXPORT CLIENTSIDE ENCRYPTION COLUMN KEY MYSCHEMA.MYCEK1, MYSCHEMA.MYCEK2 AS CSV INTO '/tmp/';
```

The example exports the information from `MY_SCHEMA TAB1` in CSV format with the REPLACE and SCRAMBLE options enabled.

```
EXPORT MY_SCHEMA.TAB1 AS CSV INTO '/tmp' WITH REPLACE SCRAMBLE THREADS 10;
```

This example exports all the database objects in schema A and B, replacing any existing export that may be present in the tmp directory.

```
EXPORT a."*", b."*" AS CSV INTO '/tmp' WITH REPLACE;
```

The following two examples show how to filter the data exported from tables by using the WHERE clause:

```
EXPORT SYSTEM.TBL_A WHERE COL_A > 5 INTO '/tmp/' WITH REPLACE;
EXPORT SYSTEM.TBL_A WHERE COL_B LIKE '%ch%' AND MOD(COL_C, 2) = 0, SYSTEM.TBL_B INTO '/tmp/' WITH REPLACE;
```

This example exports the schema `my_schema` and creates the trace file `export.trc`.

```
EXPORT my_schema."*" INTO '/usr/sap/MS1/HDB01/work' WITH TRACE 'export.trc';
```

In this example, `my_schema` has two tables, `T1` and `T2`, each with column `a`, and two views. `V1` selects `T1.a` from `T1` and `T2`. `V2` selects `*` from `T2`. The following syntax exports the object type `VIEW` and all object names not equal to `V2`. It also exports its dependency `T1`.

```
EXPORT ALL HAVING schema_name = 'my_schema' AND object_type = 'VIEW' AND object_name != 'V2' INTO '/usr/sap/MS1/HDB01/work/incexc' WITH REPLACE DEPENDENCIES object_name = 'T1';
```

This statement exports all objects of the schema `TPCH1` to the Amazon S3 bucket.

```
EXPORT TPCH1."*" AS CSV INTO 's3-eu-central-1://AKIxxxxxxxx:xl6WWxxxxxxxx@imex-demo/tpch1' WITH THREADS 4 REPLACE;
```

Related Information

- SAP HANA Database Checklists and Recommendations
- SAP HANA Security Checklists and Recommendations
- CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 741]
- M_TRACEFILE_CONTENTS System View [page 2247]
- ALTER SYSTEM REMOVE TRACES Statement (System Management) [page 564]
- IMPORT Statement (Data Import Export) [page 1029]
- M_EXPORT_BINARY_STATUS System View [page 1911]
4.10.1.126 EXPORT INTO Statement (Data Import Export)

Exports a table or view into a single CSV file.

Syntax

```
EXPORT INTO {<file_path> | <storage_path>} FROM <table_or_view_name>
[ WITH <export_into_option_list> ]
```

Syntax Elements

**file_path**

Specifies the complete path to the CSV file.

```
<file_path> ::= <string_literal>
```

**storage_path**

Specifies the cloud storage location for the export.

```
<storage_path> ::= {<azure_path> | <amazon_path> | <google_path> | <hdfs_path>}
```

**azure_path**

Specifies the location for the Azure export file.

```
<azure_path> ::= 'azure://[<azure_credentials>@]<azure_container_name>/<azure_object_id>
```

- **azure_credentials** Required when not using the WITH CREDENTIAL parameter.
  
  ```
  <azure_credentials> ::= <azure_storage_account_name>:<SAS-token>
  ```

- **azure_container_name** Specifies the name of the Azure container to access storage.

- **azure_object_id** Specifies a path to the object within the Azure storage.

**amazon_path**

SAP HANA SQL Reference Guide for SAP HANA Platform

SQL Reference

PUBLIC 1007
Specifies the location for the Amazon (AWS) export file.

```amazon_path ::= 's3-<amazon_region>://[<amazon_credentials>@]<amazon_bucket_name>/*<amazon_object_id>
```

- **amazon_region** Specifies the geographical region the bucket is located in. Refer to [Regions and Availability Zones](#).
- **amazon_credentials** Not supported with the WITH CREDENTIAL parameter. Specifies the credential key pair for API access from the AWS IAM Management Console. This is not the AWS account.

```amazon_credentials ::= '<amazon_access_key>:<amazon_secret_key>
```

- **amazon_bucket_name** Specifies the name assigned to the Amazon storage bucket when it was created.
- **amazon_object_id** Specifies a path to the object within the named bucket.

**google_path**

Specifies the location for the Google Cloud storage export file.

```google_path ::= 'gs://[<google_credentials>@]<google_bucket_name>/<google_object_id>
```

- **google_credentials** Not supported with the WITH CREDENTIAL parameter. Specifies the credential key pair for access from the Google IAM Management Console. This is not the Google Cloud account.

```google_credentials ::= <google_service_account>:<google_private_key>
```

- **google_bucket_name** Specifies the name assigned to the Google Cloud storage bucket when it was created.
- **google_object_id** Specifies a path to the object within the named bucket.

**hdlfs_path**

Specifies the location for the SAP HANA Cloud, Data Lake Files (HDLFS).

```hdlfs_path ::= hdlfs://<hdlfs_endpoint>/<hdlfs_object_id>
```

- **hdlfs_endpoint** Specifies the server address given from data lake Files. It consists of the data lake Files container id and the landscape.
- **hdlfs_object_id** Specifies a path or file name within the data lake Files container.

**table_or_view_name**

Specifies the name of the table or view to be exported.

```table_or_view_object_name ::= [ <schema_name> . ]<identifier>
```

**export_into_option_list**

Specifies a list of options that control export behavior.

```export_into_option_list ::= <export_into_option> [ <export_into_option> [ ... ] ]
```
THREADS number_of_threads

Specifies the number of threads that can be used the export. The default value is the smaller of: 10 or the number of CPU cores.

COLUMN LIST IN FIRST ROW

Indicates that the column list is stored in the first row of the CSV import file.

RECORD DELIMITED BY string_for_record_delimiter

Specifies the record delimiter of the CSV file.

FIELD DELIMITED BY string_for_field_delimiter

Specifies the field delimiter of the CSV file.

OPTIONALLY ENCLOSED BY character_for_optional_enclosure

Specifies the optional enclosure character that delimits field data.

escape_character

Specifies the escape character used in the import data.

CREDENTIAL purpose_def

Specifies the name of the credential defined in the CREATE CREDENTIAL statement. Since the credentials are defined within the credential, they no longer appear as plain text as part of export statements. The WITH CREDENTIAL clause cannot be specified when <cloud_path> contains credentials. The WITH CREDENTIAL clause is required for exports to SAP HANA Cloud, Data Lake Files, but is optional for all other cloud platforms.

Permissions

Requires:

- EXPORT system privilege
• SELECT privilege on the objects being exported.
To export column encryption keys (CEKs), you must also have the USAGE privilege on the CEK.

Description

Use the EXPORT INTO statement to export the data from a table or view into a single CSV file.
The EXPORT INTO command does not support parameterized views.

Examples

The following example creates a table and view, populates them, and exports each of them to a separate CSV file.

| CREATE TABLE t (a INT, b NVARCHAR(50));
| INSERT INTO t VALUES (0, 'test!@#$%');
| CREATE VIEW v AS SELECT * FROM t;
| EXPORT INTO '/tmp/view_content.csv' FROM v;
| EXPORT INTO '/tmp/table_with_header.csv' FROM t WITH COLUMN LIST IN FIRST ROW FIELD DELIMITED BY '!' ESCAPE '@';

This statement exports a table into CSV file in an Amazon S3 bucket.

| EXPORT INTO 's3-eu-central-1://AKIAxxxxxxxxxx:xl6WWxxxxxxxxxx@imex-demo/tpch1_lineitem.csv' FROM TPCH1.LINEITEM WITH FIELD DELIMITED BY ',', THREADS 4;

4.10.1.127 GRANT Statement (Access Control)

Grants various types of privileges to users and roles.

Syntax

| { GRANT <system_privilege> [page 0 ][{, <system_privilege>}...], TO <grantee>
| [ WITH ADMIN OPTION ]
| | GRANT <source_privilege> [page 0 ][{, <source_privilege>}...], ON REMOTE SOURCE <source_name> TO <grantee> [ WITH GRANT OPTION ]
| | GRANT <schema_privilege> [page 0 ][{, <schema_privilege>}...], ON SCHEMA <schema_name> TO <grantee> [ WITH GRANT OPTION ]
| | GRANT <object_privilege> [page 0 ][{, <object_privilege>}...], ON <object_name> TO <grantee> [ WITH GRANT OPTION ]
| | GRANT <column_key_privilege> ON CLIENTSIDE ENCRYPTION COLUMN KEY <column_encryption_key_name> TO <grantee> [ WITH GRANT OPTION ]
| | GRANT <role_name>[{, <role_name>}], TO <grantee> [ WITH ADMIN OPTION ]
| | GRANT STRUCTURED PRIVILEGE <structured_privilege> TO <grantee>
GRANT USERGROUP OPERATOR ON USERGROUP <usergroup_name> TO <grantee> [ WITH GRANT OPTION ]

Syntax Elements

grantee

Specifies the user or role that the privilege is being granted to. A role is a named collection of privileges and can be granted to either a user or another role.

<grantee> ::= <user_name> | <role_name>

<user_name> ::= <unicode_name>

<role_name> ::= [<schema_name>.]<identifier>  
<schema_name> ::= <unicode_name>

If a privilege or role is granted to a role, then all users granted that role have the specified privilege or role.

If you want to allow several database users to perform the same actions, then create a role, grant the required privileges to this role, and finally grant the role to the different database users.

When granting roles to roles, a tree of roles can be built. When granting a role (R) to a role or user (Q), user Q has all the privileges directly granted to role R and all privileges granted to roles that have been granted to role R.

<user_name> specifies the grantee’s user name.

<role_name> specifies the grantee’s role name with optional schema name.

<schema_name> specifies the schema containing the database object to be acted upon.

A reference to a schema-local role (that is, a role for which a schema_name was specified at creation time), must always be preceded by its <schema_name> wherever it is referenced (for example, GRANT <schema_name>.<role_name> TO...).

system_privilege

Grants the specified system privilege.

<system_privilege> ::= 
 { ADAPTER ADMIN
 | AGENT ADMIN
 | AGENT MESSAGING
 | ALTER CLIENTSIDE ENCRYPTION KEYPAIR
 | ATTACH DEBUGGER
 | AUDIT ADMIN
 | AUDIT OPERATOR
 | AUDIT READ
 | BACKUP ADMIN
 | BACKUP OPERATOR
 | CATALOG READ
 | CERTIFICATE ADMIN
 | CLIENT PARAMETER ADMIN
 | CREATE CLIENTSIDE ENCRYPTION KEYPAIR
 | CREATE REMOTE SOURCE
 | CREATE REMOTE SUBSCRIPTION

### System Privilege

<table>
<thead>
<tr>
<th>System Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAPTER ADMIN</td>
<td>Controls the execution of the following adapter-related statements: CREATE ADAPTER, DROP ADAPTER, and ALTER ADAPTER. It also allows access to the ADAPTERS and ADAPTER_LOCATIONS system views.</td>
</tr>
<tr>
<td>AGENT ADMIN</td>
<td>Controls the execution of the following agent-related statements: CREATE AGENT, DROP AGENT, and ALTER AGENT. It also allows access to the AGENTS and ADAPTER_LOCATIONS system views.</td>
</tr>
<tr>
<td>AGENT MESSAGING</td>
<td>Authorizes the user with which the agent communicates with the data provisioning server using HTTP protocol.</td>
</tr>
<tr>
<td>System Privilege</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ALTER CLIENTSIDE ENCRYPTION KEYPAIR</td>
<td>Authorizes a user to add a new version of a client-side encryption key pair (CKP), or to drop all older versions of the CKP.</td>
</tr>
<tr>
<td>ATTACH DEBUGGER</td>
<td>Authorizes debugging across different user sessions. For example, userA can grant ATTACH DEBUGGER to userB to allow userB to debug a procedure in userA's session (userB still needs DEBUG privilege on the procedure, however).</td>
</tr>
<tr>
<td>AUDIT ADMIN</td>
<td>Controls the execution of the following auditing-related statements: CREATE AUDIT POLICY, DROP AUDIT POLICY, and ALTER AUDIT POLICY, as well as changes to the auditing configuration. It also allows access to the AUDIT_LOG, XSA_AUDIT_LOG, and ALL_AUDIT_LOG system views.</td>
</tr>
<tr>
<td>AUDIT OPERATOR</td>
<td>Authorizes the execution of the following statement: ALTER SYSTEM CLEAR AUDIT LOG. It also allows access to the AUDIT_LOG system view.</td>
</tr>
<tr>
<td>AUDIT READ</td>
<td>Authorizes read-only access to the rows of the AUDIT_LOG, XSA_AUDIT_LOG, and ALL_AUDIT_LOG system views.</td>
</tr>
<tr>
<td>BACKUP ADMIN</td>
<td>Authorizes BACKUP and RECOVERY statements for defining and initiating backup and recovery procedures. It also authorizes changing system configuration options with respect to backup and recovery.</td>
</tr>
<tr>
<td>BACKUP OPERATOR</td>
<td>Authorizes the BACKUP statement to initiate a backup.</td>
</tr>
<tr>
<td>CATALOG READ</td>
<td>Authorizes unfiltered access to the data in the system views that a user has already been granted the SELECT privilege on. Normally, the content of these views is filtered based on the privileges of the user. CATALOG READ does not allow a user to view system views on which they have not been granted the SELECT privilege.</td>
</tr>
<tr>
<td>CERTIFICATE ADMIN</td>
<td>Authorizes the changing of certificates and certificate collections that are stored in the database.</td>
</tr>
<tr>
<td>CLIENT PARAMETER ADMIN</td>
<td>Authorizes a user to override the value of the CLIENT parameter for a database connection or to overwrite the value of the $$client$$ parameter in an SQL query.</td>
</tr>
<tr>
<td>CREATE CLIENTSIDE ENCRYPTION KEYPAIR</td>
<td>Authorizes a user to create client-side encryption key pairs.</td>
</tr>
<tr>
<td><strong>System Privilege</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CREATE REMOTE SOURCE</td>
<td>Authorizes the creation of remote data sources by using the CREATE REMOTE SOURCE statement.</td>
</tr>
<tr>
<td>CREATE REMOTE SUBSCRIPTION</td>
<td>Authorizes the creation of remote subscriptions executed on this source entry.</td>
</tr>
<tr>
<td>CREATE SCENARIO</td>
<td>Controls the creation of calculation scenarios and cubes (calculation database).</td>
</tr>
<tr>
<td>CREATE SCHEMA</td>
<td>Authorizes the creation of database schemas using the CREATE SCHEMA statement.</td>
</tr>
<tr>
<td>CREATE STRUCTURED PRIVILEGE</td>
<td>Authorizes the creation of structured (analytic privileges). Only the owner of the privilege can further grant or revoke that privilege to other users or roles.</td>
</tr>
<tr>
<td>CREDENTIAL ADMIN</td>
<td>Authorizes the use of the statements CREATE CREDENTIAL, ALTER CREDENTIAL, and DROP CREDENTIAL.</td>
</tr>
<tr>
<td>DATA ADMIN</td>
<td>Authorizes reading all data in the system views. It also enables execution of Data Definition Language (DDL) statements in the SAP HANA database.</td>
</tr>
<tr>
<td></td>
<td>A user with this privilege cannot select or change data in stored tables for which they do not have access privileges, but they can drop tables or modify table definitions.</td>
</tr>
<tr>
<td>DATABASE ADMIN</td>
<td>Authorizes all statements related to tenant databases, such as CREATE, DROP, ALTER, RENAME, BACKUP, and RECOVERY.</td>
</tr>
<tr>
<td>DATABASE START</td>
<td></td>
</tr>
<tr>
<td>DATABASE STOP</td>
<td>Authorizes a user to stop any database in the system and to select from the M_DATABASES view.</td>
</tr>
<tr>
<td>DROP CLIENTSIDE ENCRYPTION KEYPAIR</td>
<td>Authorizes a user to start any database in the system and to authorize a user to drop other users’ client-side encryption key pairs.</td>
</tr>
<tr>
<td>System Privilege</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENCRYPTION_ROOT_KEY_ADMIN</td>
<td>Authorizes all statements related to management of root keys:</td>
</tr>
<tr>
<td></td>
<td>Allows access to the system views pertaining to encryption (for example, EN-</td>
</tr>
<tr>
<td></td>
<td>CRYPTION_ROOT_KEYS, M_ENCRYPTION_OVERVIEW, M_PERSISTENCE_ENCRYPTION_STATUS,</td>
</tr>
<tr>
<td></td>
<td>M_PERSISTENCE_ENCRYPTION_KEYS, and so on).</td>
</tr>
<tr>
<td>EXPORT</td>
<td>Authorizes EXPORT to a file on the SAP HANA server. The user must also have</td>
</tr>
<tr>
<td></td>
<td>the SELECT privilege on the source tables to be exported.</td>
</tr>
<tr>
<td>EXTENDED_STORAGE_ADMIN</td>
<td>Authorizes the management of SAP HANA dynamic tiering and the creation of</td>
</tr>
<tr>
<td></td>
<td>extended storage.</td>
</tr>
<tr>
<td>IMPORT</td>
<td>Authorizes the import activity in the database using the IMPORT statements.</td>
</tr>
<tr>
<td></td>
<td>Additional privileges may also be required to be able to execute an IMPORT.</td>
</tr>
<tr>
<td></td>
<td>See the IMPORT statement for more information.</td>
</tr>
<tr>
<td>INIFILE_ADMIN</td>
<td>Authorizes making changes to system settings.</td>
</tr>
<tr>
<td>LDAP_ADMIN</td>
<td>Authorizes the use of the CREATE</td>
</tr>
<tr>
<td>LICENSE_ADMIN</td>
<td>Authorizes the use of the SET SYSTEM LICENSE statement to install a new</td>
</tr>
<tr>
<td></td>
<td>license.</td>
</tr>
<tr>
<td>LOG_ADMIN</td>
<td>Authorizes the use of the ALTER SYSTEM LOGGING [ON</td>
</tr>
<tr>
<td></td>
<td>or disable the log flush mechanism.</td>
</tr>
<tr>
<td>MONITOR_ADMIN</td>
<td>Authorizes the use of the ALTER SYSTEM statements for events.</td>
</tr>
<tr>
<td>OPTIMIZER_ADMIN</td>
<td>Authorizes the use of the ALTER SYSTEM statements concerning SQL PLAN CACHE</td>
</tr>
<tr>
<td></td>
<td>and ALTER SYSTEM UPDATE STATISTICS statements, which influence the behavior</td>
</tr>
<tr>
<td></td>
<td>of the query optimizer.</td>
</tr>
<tr>
<td>PARTITION_ADMIN</td>
<td>Authorizes the use of all non-destructive partitioning operations when</td>
</tr>
<tr>
<td></td>
<td>altering a table.</td>
</tr>
<tr>
<td>PROCESS_REMOTE_SUBSCRIPTION_EXCEPTION</td>
<td>Authorizes processing exceptions on this source entry.</td>
</tr>
<tr>
<td>System Privilege</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RESOURCE ADMIN</td>
<td>Authorizes statements concerning system resources (for example, the ALTER SYSTEM RECLAIM DATAVOLUME and ALTER SYSTEM RESET MONITORING VIEW statements). It also authorizes use of the Kernel Profiler statements, and many of the statements available in the Management Console.</td>
</tr>
<tr>
<td>ROLE ADMIN</td>
<td>Authorizes the creation and deletion of roles by using the CREATE ROLE and DROP ROLE statements. It also authorizes the granting and revoking of roles by using the GRANT and REVOKE statements. Activated repository roles, meaning roles whose creator is the predefined user _SYS_REPO, can neither be granted to other roles or users nor dropped directly. Not even users with the ROLE ADMIN privilege can do so. Check the documentation concerning activated objects.</td>
</tr>
<tr>
<td>SAVEPOINT ADMIN</td>
<td>Authorizes the execution of a savepoint using the ALTER SYSTEM SAVEPOINT statement.</td>
</tr>
<tr>
<td>SCENARIO ADMIN</td>
<td>Authorizes all calculation scenario-related activities (including creation).</td>
</tr>
<tr>
<td>SERVICE ADMIN</td>
<td>Authorizes the ALTER SYSTEM [START</td>
</tr>
<tr>
<td>SESSION ADMIN</td>
<td>Authorizes the ALTER SYSTEM commands concerning sessions to stop or disconnect a user session or to change session variables.</td>
</tr>
<tr>
<td>SQLSCRIPT LOGGING</td>
<td>Authorizes the collection of logs for a SQLScript object.</td>
</tr>
<tr>
<td>SSL ADMIN</td>
<td>Authorizes the use of the SET...PURPOSE SSL statement. It also allows access to the PSES system view.</td>
</tr>
<tr>
<td>STRUCTUREDPRIVILEGE ADMIN</td>
<td>Authorizes the creation, reactivation, and dropping of structured (analytic) privileges.</td>
</tr>
<tr>
<td>SYSTEM REPLICATION ADMIN</td>
<td>Authorizes the use of ALTER SYSTEM statements related to system replication.</td>
</tr>
<tr>
<td>System Privilege</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TABLE ADMIN</td>
<td>Authorizes the LOAD, UNLOAD and MERGE DELTA statements for tables and table partitions, as well as the ALTER TABLE statement for those clauses that do change the structure of the table and do not allow access to table data either explicitly or implicitly, for example: LOB REORGANIZE, CLEAR COLUMN JOIN DATA STATISTICS, and PRELOAD.</td>
</tr>
<tr>
<td>TRACE ADMIN</td>
<td>Authorizes the use of the ALTER SYSTEM statements related to database tracing (including the Kernel Profiler feature) and the changing of trace system settings.</td>
</tr>
<tr>
<td>TRUST ADMIN</td>
<td>Authorizes the use of statements to update the trust store.</td>
</tr>
<tr>
<td>USER ADMIN</td>
<td>Authorizes the creation and modification of users by using the CREATE</td>
</tr>
<tr>
<td>VERSION ADMIN</td>
<td>Authorizes the use of the ALTER SYSTEM RECLAIM VERSION SPACE statement of the multi-version concurrency control (MVCC) feature.</td>
</tr>
<tr>
<td>WORKLOAD ADMIN</td>
<td>Authorizes execution of the workload class and mapping statements (for example, CREATE</td>
</tr>
<tr>
<td>WORKLOAD ANALYZE ADMIN</td>
<td>Used by the Analyze Workload, Capture Workload, and Replay Workload applications when performing workload analysis.</td>
</tr>
<tr>
<td>WORKLOAD CAPTURE ADMIN</td>
<td>Authorizes access to the monitoring view M_WORKLOAD_CAPTURES to see the current status of capturing and captured workloads, as well of execution of actions with the WORKLOAD_CAPTURE procedure.</td>
</tr>
<tr>
<td>WORKLOAD REPLAY ADMIN</td>
<td>Authorizes access to the monitoring views M_WORKLOAD_REPLAY_PREPROCESSES and M_WORKLOAD_REPLAYS to see current status of preprocessing, preprocessed, replaying, and replayed workloads, as well as the execution of actions with the WORKLOAD_REPLAY procedure.</td>
</tr>
<tr>
<td>&lt;identifier&gt;.&lt;identifier&gt;</td>
<td>Components of the SAP HANA database can create new system privileges. These privileges use the component-name as the first identifier of the system privilege and the component-privilege-name as the second identifier.</td>
</tr>
</tbody>
</table>

**source_privilege**
Restricts the access and modification of a source entry.

```sql
<source_privilege> ::=  
  { CREATE VIRTUAL TABLE  
  | DROP  
  | LINKED DATABASE  
  | REMOTE EXECUTE  } 
```

**CREATE VIRTUAL TABLE**

Authorizes the creation of proxy tables pointing to remote tables from the source entry.

Proxy tables are created in a schema and point to remote entries found in a source.

**DROP**

Authorizes the DROP REMOTE SOURCE statement for the related source entry.

**LINKED DATABASE**

Authorizes execution of linked database queries.

**REMOTE EXECUTE**

Authorizes the use of the REMOTE_EXECUTE_QUERY function on the specified remote sources.

**schema_privilege**

Allows access to and modifications on a schema and the objects stored in the schema.

```sql
<schema_privilege> ::=  
  { ALL PRIVILEGES  
  | ALTER  
  | CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN  
  | CREATE ANY  
  | CREATE TEMPORARY TABLE  
  | CREATE VIRTUAL PACKAGE  
  | DEBUG  
  | DEBUG MODIFY  
  | DELETE  
  | DROP  
  | EXECUTE  
  | INDEX  
  | INSERT  
  | SELECT  
  | SELECT CDS METADATA  
  | SELECT METADATA  
  | TRIGGER  
  | UNMASKED  
  | UPDATE  } 
```

The following schema privileges are defined:

**ALL PRIVILEGES**

Grants all existing schema privilege to `<grantee>` with the exception of DEBUG, DEBUG MODIFY, and SQLSCRIPT LOGGING.

Additional privileges added later to the schema must be granted separately, or by executing another GRANT ALL PRIVILEGES statement.

**ALTER**

Allows the modification of all kinds of objects in a schema.

**CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN**
Allows a user to administer column encryption keys (CEKs), including creating, altering and dropping them. Key administrators must have the CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege.

Read access to CEKs is determined by whether a client has a CEK-copy encrypted by a public key of client key pair that is available to them in their local hdbkeystore.

**CREATE ANY**

Allows the creation of all kinds of objects, such as tables, views, sequences, synonyms, triggers, SQLScript functions, graph workspaces, or database procedures, in a schema.

**CREATE TEMPORARY TABLE**

Allows you to create a temporary local table, which can be used as input for procedures, even if the user does not have the CREATE ANY privilege for the corresponding schema.

**CREATE VIRTUAL PACKAGE**

Allows creation of virtual packages for objects, such as functions and procedures, that can be run on remote sources.

**DELETE, DROP, EXECUTE, INDEX, INSERT, SELECT, UPDATE**

The specified privilege is granted on every object stored in the specified schema currently and going forward. For detailed descriptions of the privileges, including which privileges are applicable to different objects, see the table describing object privileges.

**DEBUG**

Allows the use of debug features.

**DEBUG MODIFY**

For internal use only.

**SELECT CDS METADATA**

Allows the selection of CDS metadata from the catalog.

**SELECT METADATA**

Allows the selection of metadata for one schema from the catalog. This privilege includes access to the object definition in the case of a view, procedure, function, or trigger that may be located in different schemas.

**TRIGGER**

Allows you to create, alter, drop, enable, and disable triggers.

**UNMASKED**

Authorizes access to masked data in tables and user-defined views. This privilege is required to view the original data in views and tables that are defined with the WITH MASK clause.

Object privilege

Restricts access to, and modification of, database objects.

```
<object_privilege> ::=     { ALL PRIVILEGES
  | ALTER
  | CREATE ANY
  | CREATE OBJECT STRUCTURED PRIVILEGE
  | CREATE VIRTUAL FUNCTION
  | CREATE VIRTUAL PACKAGE
  | CREATE VIRTUAL PROCEDURE
  | CREATE VIRTUAL TABLE
```
For synonyms, the same restrictions apply to the synonym as they do for the object that the synonym represents.

The following table describes the supported object privileges in an SAP HANA database.

<table>
<thead>
<tr>
<th>Object Privilege</th>
<th>Command Types</th>
<th>Applies to</th>
<th>Privilege Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL PRIVILEGES</td>
<td>DDL &amp; DML</td>
<td>• Schemas</td>
<td>This privilege is a collection of all Data Definition Language (DDL) and Data Manipulation Language (DML) privileges that the grantor currently possesses and is allowed to grant further. The privilege it grants is specific to the particular object being acted upon. This privilege collection is dynamically evaluated for the given grantor and object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Views</td>
<td></td>
</tr>
<tr>
<td>ALTER</td>
<td>DDL</td>
<td>• Schemas</td>
<td>Authorizes the ALTER statement for the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Views</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functions/procedures</td>
<td></td>
</tr>
<tr>
<td>CREATE ANY</td>
<td>DDL</td>
<td>• Schemas</td>
<td>Authorizes all CREATE statements for the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Views</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sequences</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functions/procedures</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Remote sources</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Graph workspaces</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Triggers</td>
<td></td>
</tr>
<tr>
<td>Object Privilege</td>
<td>Command Types</td>
<td>Applies to</td>
<td>Privilege Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------</td>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CREATE OBJECT STRUCTURED PRIVILEGE</td>
<td>DDL</td>
<td>• Schemas</td>
<td>Authorizes creation of structured privilege commands on the object even if the user does not need to have the CREATE STRUCTURED PRIVILEGE.</td>
</tr>
<tr>
<td>CREATE VIRTUAL FUNCTION</td>
<td>DDL</td>
<td>• Remote sources</td>
<td>Authorizes creation of virtual functions (the REFERENCES privilege is also required).</td>
</tr>
<tr>
<td>CREATE VIRTUAL PROCEDURE</td>
<td>DDL</td>
<td>• Remote sources</td>
<td>Authorizes creation of virtual procedure to create and run procedures on a remote source.</td>
</tr>
<tr>
<td>CREATE VIRTUAL PACKAGE</td>
<td>DDL</td>
<td>• Schemas</td>
<td>Authorizes creation of virtual packages that can be run on remote sources.</td>
</tr>
<tr>
<td>CREATE VIRTUAL TABLE</td>
<td>DDL</td>
<td>• Remote sources</td>
<td>Authorizes the creation of proxy tables pointing to remote tables from the source entry.</td>
</tr>
<tr>
<td>CREATE TEMPORARY TABLE</td>
<td>DDL</td>
<td>• Schemas</td>
<td>Authorizes the creation of a temporary local table, which can be used as input for procedures, even if the user does not have the CREATE ANY privilege for the schema.</td>
</tr>
<tr>
<td>DEBUG</td>
<td>DML</td>
<td>• Schemas</td>
<td>Authorizes debug functionality for the procedure or calculation view or for the procedures and calculation views of a schema.</td>
</tr>
<tr>
<td>DEBUG MODIFY</td>
<td>DDL</td>
<td>• Functions/procedures</td>
<td>For internal use only.</td>
</tr>
<tr>
<td>Object Privilege</td>
<td>Command Types</td>
<td>Applies to</td>
<td>Privilege Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>DELETE</td>
<td>DML</td>
<td>• Schemas</td>
<td>Authorizes the DELETE and TRUNCATE statements for the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td>While DELETE applies to views, it only applies to updatable views (that is, views that do not use a join, do not contain a UNION, and do not use aggregation).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Views</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functions/procedures</td>
<td></td>
</tr>
<tr>
<td>DROP</td>
<td>DDL</td>
<td>• Schemas</td>
<td>Authorizes the DROP statements for the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Views</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sequences</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functions/procedures</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Remote sources</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Graph workspaces</td>
<td></td>
</tr>
<tr>
<td>EXECUTE</td>
<td>DML</td>
<td>• Schemas</td>
<td>Authorizes the execution of a SQLScript function or a database procedure by using the CALLS or CALL statement respectively. It also allows a user to execute a virtual function.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functions/procedures</td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>DDL</td>
<td>• Schemas</td>
<td>Authorizes the creation, modification, or dropping of indexes for the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td></td>
</tr>
<tr>
<td>INSERT</td>
<td>DML</td>
<td>• Schemas</td>
<td>Authorizes the INSERT statement for the object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td>The INSERT and UPDATE privilege are both required on the object to allow the REPLACE and UPSERT statements to be used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Views</td>
<td>While INSERT applies to views, it only applies to updatable views (views that do not use a join, do not contain a UNION, and do not use aggregation).</td>
</tr>
<tr>
<td>Object Privilege</td>
<td>Command Types</td>
<td>Applies to</td>
<td>Privilege Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>DDL</td>
<td>• Schemas</td>
<td>Authorizes the usage of all tables in this schema or this table in a foreign key definition, or the usage of a personal security environment (PSE). It also allows a user to reference a virtual function package.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td></td>
</tr>
<tr>
<td>REMOTE TABLE ADMIN</td>
<td>DDL</td>
<td>• Remote sources</td>
<td>Authorizes the creation of tables on a remote source object.</td>
</tr>
<tr>
<td>SELECT</td>
<td>DML</td>
<td>• Schemas</td>
<td>Authorizes the SELECT statement for the object or the usage of a sequence. When selection from system-versioned tables, users must have SELECT on both the table and its associated history table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Views</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sequences</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Graph workspaces</td>
<td></td>
</tr>
<tr>
<td>SELECT CDS METADATA</td>
<td>DML</td>
<td>• Schemas</td>
<td>Authorizes access to CDS metadata from the catalog.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td></td>
</tr>
<tr>
<td>SELECT METADATA</td>
<td>DML</td>
<td>• Schemas</td>
<td>Authorizes access to the complete metadata of all objects in a schema (including procedure and view definitions), including objects that may be located in other schemas.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td></td>
</tr>
<tr>
<td>TRIGGER</td>
<td>DDL</td>
<td>• Schemas</td>
<td>Authorizes the CREATE/ALTER/DROP/ENABLE and DISABLE TRIGGER statements for the specified table or the tables in the specified schema.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tables</td>
<td></td>
</tr>
<tr>
<td>Object Privilege</td>
<td>Command Types</td>
<td>Applies to</td>
<td>Privilege Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>UNMASKED</td>
<td>DML</td>
<td>Schemas, Views, Tables</td>
<td>Authorizes access to masked data in user-defined views and tables. This privilege is required to view the original data in views and tables that are defined by using the WITH MASK clause.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>DML</td>
<td>Schemas, Tables, Views</td>
<td>While UPDATE applies to views, it only applies to updatable views (views that do not use a join, do not contain a UNION, and do not use aggregation).</td>
</tr>
<tr>
<td>Object Privilege</td>
<td>Command Types</td>
<td>Applies to</td>
<td>Privilege Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------</td>
<td>------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>USERGROUP OPERATOR</td>
<td>DML</td>
<td>• User groups</td>
<td>Authorizes a user to change the settings for a user group, and to add and remove users to/from a user group. Users with the USERGROUP OPERATOR privilege can also create and drop users, but only within the user group they have the USERGROUP OPERATOR privilege on (CREATE USER <code>&lt;user_name&gt;</code> SET USERGROUP <code>&lt;usergroup_name&gt;</code>). A user can have the USERGROUP OPERATOR privilege on more than one user group, and a user group can have more than one user with the USERGROUP OPERATOR privilege on it. When granting USERGROUP OPERATOR to a user group, you must include the keyword USERGROUP before the name of the user group (for example: GRANT USERGROUP OPERATOR ON USERGROUP <code>&lt;usergroup&gt;</code> TO <code>&lt;grantee&gt;</code>). This is slightly different syntax than granting USERGROUP OPERATOR to a user.</td>
</tr>
<tr>
<td>Object Privilege</td>
<td>Command Types</td>
<td>Applies to</td>
<td>Privilege Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------</td>
<td>------------</td>
<td>----------------------</td>
</tr>
<tr>
<td><code>&lt;identifier&gt;.&lt;identifier&gt;</code></td>
<td>DDL</td>
<td>Components of the SAP HANA database can create new object privileges. These privileges use the component-name as first identifier of the system privilege and the component-privilege-name as the second identifier.</td>
<td></td>
</tr>
</tbody>
</table>

**column_key_privilege**

Specifies a column key privilege. Currently only USAGE is supported.

```sql
<column_key_privilege> ::= USAGE
```

**object_name**

```sql
<object_name> ::= [
  <table_name>
  | <view_name>
  | <sequence_name>
  | <procedure_name>
  | <synonym_name>
]
```

```sql
<table_name> ::= [<schema_name>.]<identifier>
```

```sql
<view_name> ::= [<schema_name>.]<identifier>
```

```sql
<sequence_name> ::= [<schema_name>.]<identifier>
```

```sql
<procedure_name> ::= [<schema_name>.]<identifier>
```

```sql
<synonym_name> ::= [<schema_name>.]<identifier>
```

**column_encryption_key_name**

Specifies a column encryption key name for use with client-side encryption.

```sql
<column_encryption_key_name> ::= [<schema_name>.]<identifier>
```

**privilege_name**

Grants the specified privilege.

```sql
<privilege_name> ::= <identifier>
```

**WITH ADMIN OPTION** and **WITH GRANT OPTION**

Specifies that the granted privileges can be granted further by the specified user or by users with the specified role.

**structured_privilege**
Specifies a previously defined analytic privilege based on a generic structured privilege to a user or role. This analytic privilege restricts the access for read operations to certain data in Analytic, Attribute, and Calculation Views by filtering the attribute values.

Description

The GRANT statement grants privileges and structured privileges to users and roles. It also grants roles to users and other roles.

The default schema for a user always has the same schema name as the user.

Permissions

To use the GRANT statement to grant privileges to other users and roles, a user must have the privilege and also the permissions required to grant that privilege.

SYSTEM, non-RESTRICTED users, and RESTRICTED users: A SYSTEM user has all system privileges and the role PUBLIC. However, a SYSTEM user cannot select or change data in another user's tables unless this privilege has been explicitly granted. A non-RESTRICTED user has the PUBLIC role, has the privileges required to create objects in their own default schema, and can grant privileges on their objects to other users and roles. A RESTRICTED user does not have the PUBLIC role and cannot create objects in their own schemas. RESTRICTED users have only the privileges that have been explicitly granted to them.

Users can not grant privileges to themselves.

For objects that are dependent on other objects, like views being dependent on tables, there may be times when the owner of the dependent object does not have a complete set of privileges. This situation can occur if the user does not have privileges on the underlying objects on which their object depends.

Examples

Create a schema called `my_schema`.

```sql
CREATE SCHEMA my_schema;
```

Create a table named `work_done` in the `my_schema` schema.

```sql
CREATE ROW TABLE my_schoolwork_done (t TIMESTAMP, user NVARCHAR (256), work_done VARCHAR (256));
```

Create a new user named worker with password `His_Password_1`.

```sql
CREATE USER worker PASSWORD His_Password_1;
```

Create a role called `role_for_work_on_my_schema`.

```sql
CREATE ROLE role_for_work_on_my_schema;
```
Grant the SELECT privilege on any object in `my_schema` to the `role_for_work_on_my_schema`.

```
GRANT SELECT ON SCHEMA my_schema TO role_for_work_on_my_schema;
```

Grant the INSERT privilege for the `my_schoolwork_done` table to the `role_for_work_on_my_schema`.

```
GRANT INSERT ON my_schoolwork_done TO role_for_work_on_my_schema;
```

Grant the `role_for_work_on_my_schema` role to the worker user.

```
GRANT role_for_work_on_my_schema TO worker WITH ADMIN OPTION;
```

Grant the DELETE privilege for the `my_schoolwork_done` table to the `worker` user.

```
GRANT DELETE ON my_schoolwork_done TO worker;
```

Grant the `worker` user the privilege to create any kind of object in the schema named `my_schema`.

```
GRANT CREATE ANY ON SCHEMA my_schema TO worker;
```

The result of the above statements is that the worker user has the privilege to SELECT all tables and views in schema `my_schema`, to INSERT into and DELETE from table `my_schoolwork_done`, and to create objects in schema `my_schema`. Additionally the worker user can grant DELETE on the table `myschema.work_done` to other users and roles.

Grant the INIFILE ADMIN and TRACE ADMIN privileges to the user `worker`. You grant these privileges along with the permission for the worker user to grant them further.

```
GRANT INIFILE ADMIN, TRACE ADMIN TO worker WITH ADMIN OPTION;
```

Grant the LINKED DATABASE privilege to user `myuser1` to execute linked database queries by using the remote source `myremotesys`.

```
GRANT LINKED DATABASE ON REMOTE SOURCE myremotesys TO myuser1;
```

Grant the USAGE privilege on a column encryption key `my_cek` to user `User1`, with the ability for `User1` to grant the privilege to others:

```
GRANT USAGE ON CLIENTSIDE ENCRYPTION COLUMN KEY my_cek TO User1 WITH GRANT OPTION;
```

Grant the CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege to user `KeyAdministrator` to allow to manage CEKs on schema `MySchema`:

```
GRANT CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN ON SCHEMA MySchema TO KeyAdministrator;
```

**Related Information**

- [REVOKE Statement (Access Control)](page 1098)
- [USERGROUPS System View](page 1696)
- [CLIENTSIDE_ENCRYPTION_COLUMN_KEYS System View](page 1517)
4.10.1.128 IMPORT Statement (Data Import Export)

Imports catalog objects.

Syntax

```sql
IMPORT <import_object_name_list>    
[ HAVING <object_condition> ] 
[ AS <format_option> ] 
FROM { <path> | <archive_file_name> | <storage_path> }   
[ WITH <import_option_list> ]    
[ AT [ LOCATION ] <indexserver_host_port> ]   
[ IGNORE NUMA NODE ]
```

Syntax Elements

import_object_name_list

Specifies the list of objects to be imported. The ALL and * options import all objects from a specified schema.

```sql
<import_object_name_list> ::=    <import_object_name>,<import_object_name>[,...]   | CLIENTSIDE ENCRYPTION COLUMN KEY <encryption_column_key_name>,<encryption_column_key_name>[,...]   | { ALL | * } 
```

import_object_name

Specifies objects to import data from. Specify <schema_name>.* to select all objects within a schema for import. Specify ALL (without <schema_name>) to select all objects from all schemas in the system for export. <schema_name>.* is not supported with the HAVING clause.

```sql
<import_object_name> ::= [ <schema_name>.](<identifier> | "*" | ALL)
```

If you use SAP_TIMEZONE_DATASET as the <format-option> for the AS clause, then you can only specify * or ALL for <import_object_name_list>.
encryption_column_key_name

Specifies the name of a column encryption key (CEK).

<encryption_column_key_name> ::= [ <schema_name>.]<identifier> }

HAVING

Imports a subset of data from a specified object type.

HAVING <object_condition>

The HAVING clause cannot be used in conjunction with <schema> "*".

For the types of conditions you can specify, see <condition_list>.

condition_list

Specifies the conditions defined for the HAVING and DEPENDENCIES clauses.

<condition_list> ::=  <condition> OR <condition>  | <condition> AND <condition>  
                     | NOT <condition>  
                     | ( <condition> )  | ...  <comparison_predicate> ::=  <column_expression> { = | != | <> | > | < | >= | <= } [ ANY | SOME | ALL ]  
                     | ( { <expression_list>  | <subquery> } )  <range_predicate> ::=  <column_expression> [NOT] BETWEEN <expression> AND <expression>  
                     | <in_predicate>  
                     | <like_predicate>  
                     | <null_predicate>  
<comparison_predicate> ::= [NOT] BETWEEN <expression> AND <expression>
<in_predicate> ::= [NOT] IN ( { <expression_list>  | <subquery> } )
<like_predicate> ::= [NOT] LIKE <expression> [ ESCAPE <expression> ]
<null_predicate> ::= [NOT] NULL
<expression_list> ::= <expression>[, <expression>[,...] ]
<expression> ::= <identifier> | <type_name> | <expression>
<type_name> ::= TABLE | VIEW | PROCEDURE | FUNCTION | LIBRARY | SEQUENCE | SYNONYM | TASK | GRAPH WORKSPACE | EPMMODEL | EPMQUERYSOURCE | DATA STATISTICS | ANALYTIC_PRIVILEGE | CLIENTSIDE_ENCRYPTION_COLUMN_KEY

AS format_option

Specifies the file type of the input source. CSV and BINARY are automatically detected.

<format_option> ::= CSV | BINARY [ <binary_type> ] | SHAPEFILE | LOAD_HISTORY | SAP_TIMEZONE_DATASET

If <binary_type> is not specified, then RAW (default) is used.

Specify LOAD_HISTORY to import load history traces; You can only specify one <import_object_name>
when doing so.

FROM path [ archive_file_name ]
Specifies the location where the import source is found. Specify `<archive_file_name>` if the import data is in an archive file. The archive file must have the file extension `.tar.gz` or `.tgz`.

```
<path> [ <archive_file_name> ] ::= <string_literal>
```

If AS LOAD_HISTORY is specified, then `<path>` must point to a directory that contains the unzipped dump created with `fullSystemInfoDump.py` or directly to a load history file.

If AS SHAPEFILE is specified, then `<path>` must include the directory plus the base name of the shapefile(s) (there are usually 3 files). For example, `/path/to/data/myshape`. Refer to the SAP HANA Spatial Reference guide for a complete documentation on importing from shapefiles.

If AS SAP.TIMEZONE_DATASET is specified, then `<path>` must point to a directory that contains the `tzdata.dat` file that was downloaded from `198411`\(\text{\textregistered}\), or a similarly formatted file. If no import schema is specified, then the default timezone table schema is used.

In distributed systems, `<path>` must be a shared disk; otherwise, the operation may fail or have unexpected results.

FROM `storage_path`

Specifies the cloud platform for the import.

```
<storage_path> ::=     { <azure_path>      | <amazon_path>     | <google_path>     | <hdlfs_path> }
```

azure_path

Specifies the location for the Azure import file.

```
<azure_path> ::=   'azure://<azure_credentials><azure_container_name>/<azure_object_id>'
```

azure_credentials Not supported when using the WITH CREDENTIAL parameter.

```
<azure_credentials> ::=   <azure_storage_account_name>:<SAS-token>@
```

azure_container_name Specifies the Azure credentials to access storage.

azure_object_id Specifies a path within the Azure storage.

amazon_path

Specifies the location for the Amazon (AWS) import file.

```
<amazon_path> ::=   's3-<amazon_region>://<amazon_credentials><amazon_bucket_name>/<amazon_object_id>'
```

amazon_region Specifies the geographical region the bucket is located in. Refer to Regions and Availability Zones\(\text{\textregistered}\).

amazon_credentials Specifies the credential key pair for API access from the AWS IAM Management Console. This is not the AWS account. Not supported with the WITH CREDENTIAL parameter.

```
<amazon_credentials> ::=   
```
amazon_access_key:amazon_secret_key

- **amazon_bucket_name**: Specifies the name assigned to the bucket when it was created.
- **amazon_object_id**: Specifies a path within the named bucket.

**google_path**

Specifies the location for the Google Cloud storage import file.

<google_path> ::= 'gs://<google_credentials><google_bucket_name>/<google_object_id>'

- **google_credentials**: Not supported with the WITH CREDENTIAL parameter. Specifies the credential key pair for access from the Google IAM Management Console. This is not the Google Cloud account.

- **google_bucket_name**: Specifies the name assigned to the Google Cloud storage bucket when it was created.
- **google_object_id**: Specifies a path to the object within the named bucket.

**hdlfs_path**

Specifies the location for the SAP HANA Cloud, Data Lake Files (HDLFS).

<hdlfs_path> ::= hdlfs://<hdlfs_endpoint>/<hdlfs_object_id>

- **hdlfs_endpoint**: Specifies the server address given from data lake Files. It consists of the data lake Files container id and the landscape.
- **hdlfs_object_id**: Specifies a path or file name within the data lake Files container. For example, hdlfs://example-file-container.files.hdl.hc-XXX.XXX.hanacloud.ondemand.com/directory/sample.dat

**WITH import_option_list**

Specifies a list of import options. If AS LOAD_HISTORY or AS SAP_TIMEZONE_DATASET is also specified, then WITH REPLACE is the only supported import option.

<import_option_list> ::= <import_option>[, <import_option>[,...]]
<import_option> ::= REPLACE,
                  CATALOG ONLY,
                  DATA ONLY,
                  NO DEPENDENCIES,
                  DEPENDENCIES <object_condition>,
                  THREADS <number_of_threads>,
                  RENAME SCHEMA <rename_schema_list>,
                  FAIL ON INVALID DATA,
                  IGNORE EXISTING,
                  PERSISTENT MEMORY,
                  STATISTICS ONLY,
                  NO STATISTICS,
                  LOOPBACK REMOTE SOURCE,
                  RENAME REMOTE OBJECT,
                  <shapefile_option_list>,
                  TRACE,
                  CREDENTIAL '<purpose_def>'

- **REPLACE**:
Defines the behavior if the import data already exists in the database. When specified, if a table defined in the import data currently exists in the database, then it is dropped and recreated before the data is imported. If the REPLACE option is not specified, then an error is thrown if an existing database table is defined in the import data.

**CATALOG ONLY**

Imports only the database catalog.

This option interacts with the STATISTICS ONLY and NO STATISTICS options.

**DATA ONLY**

Specifies that only the data in the import file should be imported, without updating or changing the metadata. This option is only valid when the specified target object has its own data, such as a table or a data statistics (as opposed to procedures and views, whose data is materialized from other objects at runtime). The target objects must exist in the database and their definition must match that of the data being imported.

When specifying `IMPORT <schema_name>.*` and `IMPORT ALL...` with DATA ONLY, other objects that do not have records are ignored.

For BINARY data, the existing table data is overwritten with the imported data. For CSV data without the REPLACE option, the import data is appended to the existing table data.

This option interacts with the STATISTICS ONLY and NO STATISTICS option and is not supported for virtual tables.

**NO DEPENDENCIES**

Specifies to not import the underlying dependencies of an import object.

**DEPENDENCIES**

Specifies the underlying dependencies of an object to import.

```
DEPENDENCIES <object_condition>
```

For the types of conditions you can specify, see `<condition_list>`.

**THREADS number_of_threads**

Specifies how many objects are imported in parallel; the default is 1.

```
<number_of_threads> ::= <unsigned_integer>
```

Increasing the number of threads reduces import time, but can also negatively affect database system performance. Consider these items when you use this parameter:

- When importing a single table, THREADS has no effect.
- When importing a view or procedure, two or more threads should be used, up to the number of dependent objects.
- When importing a whole schema consider using more than 10 threads. With a maximum being the number of CPU cores in the system.
- When importing a whole BW/ERP system database with tens of thousands of tables by using the ALL keyword, a large number of threads can be used (up to 256).

**RENAME SCHEMA rename_schema_list**
Specifies whether to rename the object’s schema during the import.

```plaintext
<rename_schema_list> ::= <rename_schema_token>
[,<rename_schema_token>[],]<rename_schema_token>
<rename_schema_token> ::= <source_schema> TO <target_schema>
```

You cannot specify the same schema as both `source_schema` and `target_schema` in the same or a different `rename_schema_token`.

**FAIL ON INVALID DATA**

Fails the import operation if not all data is imported successfully.

**IGNORE EXISTING**

Does not import objects that already exist in the database. The IGNORE EXISTING option is ignored, if it is used in combination with REPLACE.

**PERSISTENT MEMORY** Imports persistent memory data files from exported data into persistent memory. If no persistent memory data files are found, then this clause is ignored. If this option is specified and persistent memory is not configured for the database, then the IMPORT statement fails.

**STATISTICS ONLY**

Data statistics objects are imported by default, along with the objects they reference in `import_object_name_list` (for example, tables). When you specify STATISTICS ONLY, the metadata and data for non-statistics objects in `import_object_name_list` are excluded from the import.

If you specify STATISTICS ONLY without DATA ONLY or CATALOG ONLY, then the data and metadata for data statistics objects are imported.

If you specify STATISTICS ONLY and DATA ONLY, then only the data for the data statistics objects is imported.

If you specify STATISTICS ONLY and CATALOG ONLY, then only the metadata for the data statistics objects is imported.

**NO STATISTICS**

Excludes data statistics objects from the import. Metadata and data for non-statistics objects in `import_object_name_list` are still imported, and they are impacted as normal by whether CATALOG ONLY or DATA ONLY is specified.

**LOOPBACK REMOTE SOURCE** Creates a reproduction workflow for virtual tables. When importing tables, a loopback remote source is created that points to the local SAP HANA server. For every virtual table imported, a corresponding dummy local table is created with the same metadata. Virtual tables are then created by using this loopback remote source and the dummy local tables.

**RENAME REMOTE OBJECT** Specify a mapping of the old remote object to new remote object when importing virtual tables into another system.

When importing virtual tables, each virtual table is created by using the same remote source and remote object when it was exported by default.

**shapefile_option_list**

Refer to the SAP HANA Spatial Reference guide for the current, complete option list for importing shapefiles.

**TRACE import_trace_file_name**
Writers execution information to the specified trace file when importing data.

\[\text{<import_trace_file_name> ::= <string_literal>}\]

The info level trace file contains detailed error descriptions for troubleshooting. The file is saved to the current trace directory.

The contents of the trace file can be found in the M_TRACEFILE_CONTENTS system view. It contains information on:

- The single archive for the single archive import
- The import list
- Import feasibility checks
- Objects created and dropped during the import
- Schemas created during the import
- Schemas renamed during the import
- Metadata imported for column tables
- The format of the imported data (binary or CSV)
- Indexes, triggers, and foreign keys created during the import
- The results table from the import

To delete the trace file, use the ALTER SYSTEM REMOVE TRACES statement.

\textbf{CREDENTIAL purpose_def}

Specifies the name of the credential defined in the CREATE CREDENTIAL statement. Since the credentials are defined within the credential, they no longer appear as plain text as part of import statements. The WITH CREDENTIAL clause cannot be specified when \texttt{<cloud_path>} contains credentials. The WITH CREDENTIAL clause is required for imports from SAP HANA Cloud, Data Lake Files, but is optional for all other cloud platforms.

\textbf{AT LOCATION indexserver_host_port}

Specifies the index server on which tables are created and imported.

\[\text{<indexserver_host_port> ::= ',<host_name>:<port_number>}'\]

If you specify the hostname and port, then the tables are created and imported at that location.

\textbf{IGNORE NUMA NODE}

By default, SAP HANA attempts to apply the NUMA node preferences that were set in the source system for the table you are importing. If the number of NUMA nodes on the target system is greater than or equal to the number on the source system, then the NUMA node preferences are applied. If the target system has fewer NUMA nodes, then the NUMA node preferences are ignored.

Specify IGNORE NUMA NODE to override the default behavior and not import NUMA location preferences into the target system.

\section*{Permissions}

Requires the following:
• IMPORT system privilege
• CREATE SCHEMA privilege if the schema for the import object does not exist
• CREATE privilege on the schema for each object type to import
  DROP privilege on object to drop if object already exists and WITH REPLACE option is used
• CREATE TEMPORARY TABLE privilege on the current schema
• CREATE TEMPORARY TABLE privilege on the schema containing the object to export.

In addition to the IMPORT privilege, you must have CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege on the schema and CREATE CLIENTSIDE ENCRYPTION KEYPAIR privilege to import CEKs.

Description

The IMPORT statement imports catalog objects (tables, views, synonyms, sequences, and procedures) that have previously been exported with the EXPORT statement. To import external data into existing tables, use the IMPORT FROM statement. The IMPORT statement can also import load history data that is stored in nameserver_history.trc files and timezone definitions according to SAP note 198411.

The format (BINARY or CSV) of the file being imported is detected automatically.

You can only import exported binary files that were generated on an SAP HANA system with the same endianness.

System-versioned tables and their associated history table must be imported separately. After import, execute an ALTER TABLE statement to associate the system-versioned table and its history table and specify NOT VALIDATED to bypass the check for whether the history table is empty.

Cancel an import session by using the ALTER SYSTEM CANCEL WORK IN SESSION statement and specifying the connection ID from the M_IMPORT_BINARY_STATUS view.

When using CSV format to export/import tables that have columns of type GENERATED ALWAYS AS <expression>, verify that the imported records match what was exported. If there are anomalies:

• For column tables, switch to exporting/importing as BINARY.
• For row tables, change the table to be a column table and export/import as BINARY.

For client-side encryption, importing a table with encrypted columns fails if there are no CEK copies for a given CEK used by the table. Importing a CEK copy imports the corresponding public keys of client key pairs as well. Importing a CEK WITH REPLACE fails if the existing CEK encrypts a column. Importing encrypted data into an existing table requires the UUIDs of the CEKs of the exported data to match the UUIDs of the CEKs of existing table columns.

A table with encrypted columns along with the CEK copies and public keys may be successfully imported into a new database, but you cannot access the encrypted data if the corresponding private keys are also not available to the clients trying to access that data. Either the clients accessing imported encrypted column must be the same clients that accessed them before and already have the private keys, or the private keys must be exported from the old clients and imported into the new ones separately.

Detailed results of the last successful execution of the IMPORT statement are stored in the following session-local temporary table <current_schema>#IMPORT_RESULT. If IMPORT has not been executed in the current session, selecting from <current_schema>#IMPORT_RESULT returns an error. If <current_schema> of the session is invalid, then IMPORT throws an exception as it cannot create its result table for the <current_schema>.
### Example

The following example imports the contents of the `tzdata.dat` into the default timezone table schema:

```sql
IMPORT * AS SAP_TIMEZONE_DATASET FROM '/path/to/tzdata.dat;
```

The following example imports a client-side encryption key:

```sql
IMPORT CLIENTSIDE ENCRYPTION COLUMN KEY MYSHEMA.MYCEK1, MYSCHAM.MYCEK2 FROM '/path/to/cek';
```

This example imports all tables from schema TESTSCH by using the REPLACE option and ten execution threads.

```sql
IMPORT TESTSCH."*" AS CSV FROM '/tmp' WITH REPLACE THREADS 10;
```

This example imports the load history data that is stored in an unzipped dump in directory `/path/to/dump` into the tables MYSCHEMA.DATA_LOAD_HISTROY_HOST and MYSCHEMA.DATA_LOAD_HISTROY_SERVICE:

```sql
IMPORT MYSCHEMA.DATA AS LOAD_HISTORY FROM '/path/to/dump';
```

This example imports the load history data from the given `nameserver_history.trc` file into the tables MYSCHEMA.DATA_LOAD_HISTROY_HOST and MYSCHEMA.DATA_LOAD_HISTROY_SERVICE:

```sql
IMPORT MYSCHEMA.DATA AS LOAD_HISTORY FROM '/path/to/nameserver_history.trc';
```

This example imports the schema MY_SCHEMA and creates the trace file `import.trc`:

```sql
IMPORT MY_SCHEMA."*" FROM '/usr/sap/MS1/HDB01/work' WITH TRACE 'import.trc' REPLACE;
```
This example imports the schema from table T14, but ignores the NUMA NODE preferences.

```
IMPORT SYSTEM:"T14" FROM '/tmp/test/' WITH REPLACE THREADS 40 IGNORE NUMA NODE;
```

In this example, `MY_SCHEMA` has two tables, `T1` and `T2`, each with column `A`, and two views. V1 selects `T1.a` from `T1` and `T2`. V2 selects `*` from `T2`. The following syntax imports the object type `VIEW` and all object names not equal to `V2`. It also imports its dependency `T1`.

```
IMPORT ALL HAVING schema_name = 'my_schema' AND object_type = 'VIEW' AND
object_name != 'V2'
    FROM '/usr/sap/MS1/HDB01/work' WITH REPLACE DEPENDENCIES object_name = 'T1';
```

This statement imports from Amazon storage, all objects from the exported location.

```
IMPORT ALL FROM 's3-eu-central-1://AKIAxxxxxxxxxx:xl6WWxxxxxxxxxx@imex-demo/
tpch1' WITH THREADS 4 REPLACE;
```

**Related Information**

- SAP HANA Spatial Reference
- CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 741]
- M_TRACEFILE_CONTENTS System View [page 2247]
- ALTER SYSTEM REMOVE TRACES Statement (System Management) [page 564]
- EXPORT Statement (Data Import Export) [page 997]
- M_IMPORT_BINARY_STATUS System View [page 1939]
- IMPORT FROM Statement (Data Import Export) [page 1038]
- ALTER SYSTEM CANCEL [WORK IN] SESSION Statement (System Management) [page 516]

**4.10.1.129 IMPORT FROM Statement (Data Import Export)**

Imports external data from a file into an existing table or an existing data statistics object.

**Syntax**

```
IMPORT FROM [ <file_type> ] { <file_path> | <storage_path> } 
[ INTO <database_object_name> ] 
[ WITH <import_from_option_list> ]
```
Syntax Elements

**file_type**

Specifies the type of file being imported from.

```plaintext
<file_type> ::= CSV FILE | CONTROL FILE
```

When importing data statistics objects, you can only import from a control file.

If `<file_type>` is not explicitly specified, the system will treat the input file as a control file.

**file_path**

Specifies the complete path and file name of the file to import.

```plaintext
<file_path> ::= <string_literal>
```

Named pipes (FIFO) are supported for `<file_path>`.

When `<file_type>` is CSV, a GZIP (.gz) file is supported in `<file_path>` to indicate a GZIP file containing the CSV data.

In distributed systems, `<file_path>` must be a shared disk; otherwise, the operation may fail or have unexpected results.

**storage_path**

Specifies the cloud storage location for the import.

```plaintext
<storage_path> ::= { <azure_path> | <amazon_path> | <google_path> | <hdlfs_path> }
```

**azure_path**

Specifies the location for the Azure import file.

```plaintext
<azure_path> ::= 'azure://<azure_credentials><azure_container_name>/<azure_object_id>'
```

- **azure_credentials** Not supported when using the WITH CREDENTIAL parameter.

```plaintext
<azure_credentials> ::= <azure_storage_account_name>:<SAS-token>@
```

- **azure_container_name** Specifies the Azure credentials to access storage.
- **azure_object_id** Specifies a path within the Azure storage.

**amazon_path**

Specifies the location for the Amazon (AWS) import file.

```plaintext
<amazon_path> ::= 's3-<amazon_region>://<amazon_credentials><amazon_bucket_name>/
<amazon_object_id>'
```

- **amazon_region** Specifies the geographical region the bucket is located in. Refer to Regions and Availability Zones.
amazon_credentials Specifies the credential key pair for API access from the AWS IAM Management Console. This is not the AWS account. Not supported with the WITH CREDENTIAL parameter.

```
<amazon_credentials> ::= <amazon_access_key>:<amazon_secret_key>@
```

amazon_bucket_name Specifies the name assigned to the bucket when it was created.

amazon_object_id Specifies a path within the named bucket.

google_path

Specifies the location for the Google Cloud storage import file.

```
<google_path> ::= 'gs://<google_credentials><google_bucket_name>/<google_object_id>'
```

google_credentials Not supported with the WITH CREDENTIAL parameter. Specifies the credential key pair for access from the Google IAM Management Console. This is not the Google Cloud account.

```
<google_credentials> ::= <google_service_account>:<google_private_key>@
```

google_bucket_name Specifies the name assigned to the Google Cloud storage bucket when it was created.

google_object_id Specifies a path to the object within the named bucket.

hdlfs_path

Specifies the location for the SAP HANA Cloud, Data Lake Files (HDLFS).

```
<hdlfs_path> ::= hdlfs://<hdlfs_endpoint>/<hdlfs_object_id>
```

hdlfs_endpoint Specifies the server address given from data lake Files. It consists of the data lake Files container id and the landscape.

hdlfs_object_id Specifies a path or file name within the data lake Files container. For example, hdlfs://example-file-container.files.hdl.hc-XXX.XXX.hanacloud.ondemand.com/directory/sample.dat

database_object_name

Specifies the target table name or data statistics object name into which the imported data will be imported.

```
<database_object_name> ::= [
    <schema_name>.]<table_name> |
    STATISTICS  [<schema_name>.]<data_statistics_object_name>
<schema_name> ::= <unicode_name>
<table_name> ::= <identifier>
<data_statistics_object_name> ::= <identifier>
```

Specify STATISTICS when importing data into an existing database statistics object. If the data statistics objects already has data, then the data is overwritten by the data being imported. INTO STATISTICS can only be specified in a control file. When importing data to a data statistics object, the only supported <import_from_option> option is ERROR LOG.

import_from_option_list
Specifies a list of options that control import behavior.

WITH <import_from_option_list>
<import_from_option_list> ::= <import_from_option> [, ,<import_from_option> [,], ]

<import_from_option> ::= THREADS <number_of_threads> | BATCH <number_of_records_of_each_commit>
| TABLE LOCK | NO TYPE CHECK | SKIP FIRST <number_of_rows_to_skip> ROW | COLUMN LIST IN FIRST ROW [with_schema_flexibility]
| RECORD DELIMITED BY <string_for_record_delimiter>
| FIELD DELIMITED BY <string_for_field_delimiter>
| OPTIONALLY ENCLOSED BY <character_for_optional_enclosure>
| ESCAPE <escape_character>
| DATE FORMAT <string_for_date_format>
| TIME FORMAT <string_for_time_format>
| TIMESTAMP FORMAT <string_for_timestamp_format>
| ERROR LOG <file_path_of_error_log>
| FAIL ON INVALID DATA | CREDENTIAL '<purpose_def>'

THREADS <number_of_threads>

Specifies the number of threads that can be used for concurrent import. The default value is 1 and the maximum allowed value is 256.

THREADS <number_of_threads>
<number_of_threads> ::= <unsigned_integer>

BATCH <number_of_records_of_each_commit>

Specifies the number of records to be inserted in each commit.

BATCH <number_of_records_of_each_commit>
<number_of_records_of_each_commit> ::= <unsigned_integer>

THREADS and BATCH provide high loading performance by enabling parallel loading and also by committing many records at once. In general, for column tables, a good setting to use is 10 parallel loading threads, with a commit frequency of 10,000 records or greater.

TABLE LOCK

Provides faster data loading for column store tables.

Use this option carefully as it incurs table locks in exclusive mode, as well as explicit hard merges and save points after data loading is finished. The performance gain from this option can vary according to the table constraints (like primary key) and optimization of other layers (like persistence or a DML command).

NO TYPE CHECK

Specifies that the record is inserted without checking the type of each field.

SKIP FIRST <number_of_rows_to_skip> ROW

Skips the specified number of rows in the import file.

SKIP FIRST <number_of_rows_to_skip> ROW
<number_of_rows_to_skip> ::= <unsigned_integer>

COLUMN LIST IN FIRST ROW [with_schema_flexibility]
Indicates that the column list is stored in the first row of the CSV import file.

```
COLUMN LIST IN FIRST ROW [<with_schema_flexibility>]
<with_schema_flexibility> ::= WITH SCHEMA FLEXIBILITY
```

WITH SCHEMA FLEXIBILITY creates missing columns in flexible tables during CSV imports, as specified in the header (first row) of the CSV file or column list. By default, missing columns in flexible tables are not created automatically during data imports.

```
COLUMN LIST ( column_name_list ) [with_schema_flexibility]
```

Specifies the list of table columns for the data being imported.

```
COLUMN LIST (<column_name_list>)
<column_name_list> ::= <column_name> [, <column_name>...]
<column_name> ::= <identifier>
<with_schema_flexibility> ::= WITH SCHEMA FLEXIBILITY
```

The name list has one or more column names. The ordering of the column names has to match the order of the column data in the CSV file from the leftmost column. The column names must exist in the target table, and the data types must match those in the CSV file. Unmatched columns in the target table have NULL values. If the number of column names is larger than the number of columns in the CSV file, unmatched columns have NULL values. WITH SCHEMA FLEXIBILITY creates missing columns in flexible tables during CSV imports, as specified in the header (first row) of the CSV file or column list. By default, missing columns in flexible tables are not created automatically during data imports.

```
RECORD DELIMITED BY string_for_record_delimiter and FIELD DELIMITED BY string_for_field_delimiter
```

Specifies the record and field delimiters used in the CSV file, respectively. Supported delimiters are as follows:

- Any string of ASCII characters
- \uhhhhhh: 8 hexadecimal digits
- \uhhhh: 4 hexadecimal digits
- \xhh…: 1 or more hexadecimal digits
- \nnn: one, two, or three octal numbers

Special characters such as `\036` or `\u0007` are useful when importing datafiles that have special characters like `\u0001` or chr(1). These values are then converted to non-escaped values and used as the delimiter.

```
OPTIONALLY ENCLOSED BY character_for_optional_enclosure
```

Specifies the optional enclosure character that delimits field data.

```
OPTIONALLY ENCLOSED BY <character_for_optional_enclosure>
<character_for_optional_enclosure> ::= <character_literal>
```

```
ESCAPE escape_character
```

Specifies the escape character used in the import data.

```
<escape_character> ::= <character_literal>
```

```
DATE FORMAT string_for_date_format
```

SAP HANA SQL Reference Guide for SAP HANA Platform

1042 PUBLIC

SQL Reference
Specifies the format that date strings are encoded with in the import data.

```sql
DATE FORMAT <string_for_date_format>
<string_for_date_format> ::= <string_literal>
```

Values can include:

- Y: year
- MM: month
- MON: name of month
- DD: day

For example:

- 'YYYYMMDD' = 20120520
- 'YYYY-MM-DD' = 2012-05-20
- 'YYYY-MON-DD': 2012-MAY-20

**TIME FORMAT string_for_time_format**

Specifies the format that time strings are encoded with in the import data.

```sql
TIME FORMAT <string_for_time_format>
<string_for_time_format> ::= <string_literal>
```

Values can include:

- HH24: hour
- MI: minute
- SS: second

For example:

- 'HH24MISS': 143025
- 'HH24:MI:SS': 14:30:25

**TIMESTAMP FORMAT string_for_timestamp_format**

Specifies the format that timestamp strings are encoded with in the import data.

```sql
TIMESTAMP FORMAT <string_for_timestamp_format>
<string_for_timestamp_format> ::= <string_literal>
```

For example:

- 'YYYY-MM-DD HH24:MI:SS': 2012-05-20 14:30:25

**ERROR LOG file_path_of_error_log**

Specifies that a log file of errors is generated and stored in this file. Ensure that the database can write to this file.

```sql
ERROR LOG <file_path_of_error_log>
<file_path_of_error_log> ::= <string_literal>
```

For security reasons, the path for ERROR LOG must not contain symbolic links and must not point to the same drive location (folder) as the database. It can, however, point to the location of the backup or work subfolders.

**FAIL ON INVALID DATA**
Specifies that the IMPORT FROM command fails unless all the entries import successfully.

**CREDENTIAL purpose_def**

This option is not supported with the `<file_type> JSON`.

Specifies the name of the credential defined in the CREATE CREDENTIAL statement. Since the credentials are defined within the credential, they no longer appear as plain text as part of import statements. The WITH CREDENTIAL clause cannot be specified when `<cloud_path>` contains credentials. The WITH CREDENTIAL clause is required for imports from SAP HANA Cloud, Data Lake Files, but is optional for all other cloud platforms.

**Description**

Use the IMPORT FROM statement to import data into database objects, such as tables and data statistics objects from files such as CSV files.

To import catalog objects (tables, views, and so on) that have been exported with the EXPORT statement, use the IMPORT statement instead.

When importing from data statistics objects, the schema of a data statistics object is the same as that of the schema of the data source on which the data statistics object was created.

All `<string_literals>` in the `<import_from_option>` support UTF-8 except surrogate-pair encoding.

For security reasons, only CSV files located at paths defined in the `csv_import_path_filter` configuration parameter can be loaded by using the IMPORT FROM SQL statement. This feature can be disabled by using the `enable_csv_import_path_filter` configuration parameter. Two related configuration parameters are specified in the `import_export` section of the indexserver (nameserver in the case of multi-DB) configuration, so you can turn off this feature or update path filter as follows:

```
ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'system') set
('import_export', 'enable_csv_import_path_filter') = 'false' with reconfigure
ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'system') set
('import_export', 'csv_import_path_filter') = '/A;/B' with reconfigure
```

Once you add a path `/A` to the path filter, every sub-path of `/A` is automatically added as well.

When using CSV format to export/import tables that have columns of type `GENERATED ALWAYS AS <expression>`, verify that the imported records match what was exported. If there are anomalies:

- For column tables, switch to exporting/importing as BINARY.
- For row tables, change the table to be a column table and export/import as BINARY.

**Permissions**

Requires:

- IMPORT system privilege
- INSERT privilege on the object that is being imported into.
Examples

Example - Auto data type promotion during importing from flexible tables

Suppose you have a CSV file, `data.csv`, similar to the following:

```
ORG_ID,INT_COL,DEC_COL,DATE_COL,STR_COL,LOB_COL
1,12,12.34,"2017-01-03","my text","my text is longer than 5000 chars"
```

And that you have a table defined as follows:

```
CREATE COLUMN TABLE TEST.IMPORT_TABLE (     "_ID" BIGINT GENERATED ALWAYS AS IDENTITY ) WITH SCHEMA FLEXIBILITY (DEFAULT DATA TYPE * AUTO DATA TYPE PROMOTION);
```

Your import statement would look as follows:

```
IMPORT FROM CSV FILE 'data.csv'     INTO TEST.IMPORT_TABLE     WITH THREADS 20     BATCH 50000     COLUMN LIST IN FIRST ROW WITH SCHEMA FLEXIBILITY     RECORD DELIMITED BY '
'     FIELD DELIMITED BY ','     OPTIONALLY ENCLOSED BY '"'     DATE FORMAT 'YYYY-MM-DD'     ERROR LOG 'error_log.txt' ;
```

Your resulting table columns and data types would be as follows:

```
_ID BIGINT
ORG_ID INTEGER
INT_COL INTEGER
DEC_COL DECIMAL
DATE_COL DATE
STR_COL NVARCHAR(5000)
LOB_COL NCLOB
```

Example - Importing CSV Data

Create a table, `mytable`, to store the imported data.

```
CREATE ROW TABLE mytable ( A INT, B VARCHAR(10), C DATE, D TIME, E DECIMAL );
```

Create a CSV text file, `/data/data.csv`, and add the following contents.

```
1,"DATA1","2012-05-20","14:30:25",123456
2,"DATA2","2012-05-21","15:30:25",234567
3,"DATA3","2012-05-22","16:30:25",345678
4,"DATA4","2012-05-23","17:30:25",456789
```

Execute the following command to import the data.

```
IMPORT FROM CSV FILE '/data/data.csv' INTO "MYTABLE"     WITH RECORD DELIMITED BY '
'     FIELD DELIMITED BY ',';
```

Example - Importing by using a control file
Import the data from Example 1 using a control file you create (/data/data.ctl) that points to the CSV file. The control file contains the following commands:

```
IMPORT DATA INTO TABLE "MYTABLE" FROM '/data/data.csv'
  RECORD DELIMITED BY '
'
  FIELD DELIMITED BY ','
  OPTIONALLY ENCLOSED BY '"'
  ERROR LOG '/data/data.err';
```

Execute the following command to import the data using the control file: IMPORT FROM CONTROL FILE '/data/data.ctl';

The fictitious example below imports data into the data statistics object My_Schema.SKETCH5 using a control file (/data/data.ctl). The control file contains the following commands:

```
IMPORT DATA INTO STATISTICS "My_Schema"."SKETCH5" FROM '/data/data'
  ERROR LOG '/data/data.err';
```

Execute the following command to import the data using the control file: IMPORT FROM CONTROL FILE '/data/data.ctl';

**Example - Importing by using date formats**

In the example below, the date format is of the CSV import data is different from the default date format 'YYYY-MM-DD'. In this import data, the date format used is 'MM-DD-YYYY'. You create a CSV text file, /data/data_different_date.csv, and add the following contents.

```
1,"DATA1","2012-05-20",123456
2,"DATA2","2012-05-21",234567
3,"DATA3","2012-05-22",345678
4,"DATA4","2012-05-23",456789
```

Execute the following command to import the data.

```
IMPORT FROM CSV FILE '/data/data_different_date.csv' INTO "MYTABLE" WITH RECORD DELIMITED BY '
'
  FIELD DELIMITED BY ','
  DATE FORMAT 'MM-DD-YYYY';
```

**Example - Importing by using COLUMN LIST**

Create a table called COLLIST to store the imported data.

```
CREATE ROW TABLE COLLIST ( A INT, B VARCHAR(10), C DATE, D DECIMAL );
```

Create a CSV text file /data/data_col_list.csv and add the following contents.

```
1,"DATA1",2012-05-20,123456
2,"DATA2",2012-05-21,234567
3,"DATA3",2012-05-22,345678
4,"DATA4",2012-05-23,456789
```

Execute the following commands to import the data using a column list.

```
IMPORT FROM CSV FILE '/data/data_col_list.csv' INTO "COLLIST" WITH RECORD DELIMITED BY '
'
  FIELD DELIMITED BY ','
  COLUMN LIST ("A", "B", "C", "D");
```
To import data without dealing with the proper table layout, it is possible to use WITH SCHEMA FLEXIBILITY as an extended option of COLUMN LIST to import into a flexible table.

Create a flexible table to store the imported data as follows:

```sql
CREATE ROW TABLE FLEX ( X INT ) WITH SCHEMA FLEXIBILITY;
```

Execute the following commands to import previously created CSV file `data_col_list.csv` without explicitly adding columns.

```sql
IMPORT FROM CSV FILE '/data/data_col_list.csv' INTO "FLEX"    WITH RECORD DELIMITED BY '\n'
FIELD DELIMITED BY ','
COLUMN LIST ("A", "B", "C", "D")
WITH SCHEMA FLEXIBILITY;
```

This statement imports data in a CSV file in an S3 bucket into table TPCH1.LINEITEM. The CSV file is delimited by a comma and the import uses 4 threads.

```sql
IMPORT FROM CSV FILE 's3-eu-central-1://AKIAxxxxxxxxxx:xl6WWxxxxxxxxxx@imex-demo/tpch1_lineitem.csv' INTO TPCH1.LINEITEM WITH FIELD DELIMITED BY ',' THREADS 4;
```

**Related Information**

EXPORT Statement (Data Import Export) [page 997]
IMPORT Statement (Data Import Export) [page 1029]

**4.10.1.130 IMPORT SCAN Statement (Data Import Export)**

Searches the specified path for exported objects.

**Syntax**

```sql
IMPORT SCAN { <path> | <storage_path> } [ WITH CREDENTIAL '<purpose_def>' ]
```

**Syntax Elements**

Specifies the cloud storage location for the import.

```sql
<storage_path> ::=     { <azure_path>      | <amazon_path>     | <google_path>
```
Refer to the local temporary table #IMPORT_SCAN_RESULT to check the list of tables.

azure_path

Specifies the location for the Azure import file.

```
<azure_path> ::= 'azure://<azure_credentials><azure_container_name>/<azure_object_id>'
```

azure_credentials Not supported when using the WITH CREDENTIAL parameter.

```
<azure_credentials> ::= <azure_storage_account_name>:<SAS-token>@
```

azure_container_name Specifies the Azure credentials to access storage.

azure_object_id Specifies a path within the Azure storage.

amazon_path

Specifies the location for the Amazon (AWS) import file.

```
<amazon_path> ::= 's3-<amazon_region>://<amazon_credentials><amazon_bucket_name>/<amazon_object_id>'
```

amazon_region Specifies the geographical region the bucket is located in. Refer to Regions and Availability Zones.

amazon_credentials Specifies the credential key pair for API access from the AWS IAM Management Console. This is not the AWS account. Not supported with the WITH CREDENTIAL parameter.

```
<amazon_credentials> ::= <amazon_access_key>:<amazon_secret_key>@
```

amazon_bucket_name Specifies the name assigned to the bucket when it was created.

amazon_object_id Specifies a path within the named bucket.

google_path

Specifies the location for the Google Cloud storage import file.

```
<google_path> ::= 'gs://<google_credentials><google_bucket_name>/<google_object_id>'
```

google_credentials Not supported with the WITH CREDENTIAL parameter. Specifies the credential key pair for access from the Google IAM Management Console. This is not the Google Cloud account.

```
<google_credentials> ::= <google_service_account>:<google_private_key>@
```

google_bucket_name Specifies the name assigned to the Google Cloud storage bucket when it was created.

google_object_id Specifies a path to the object within the named bucket.

dlf_path

Specifies the location for the SAP HANA Cloud, Data Lake Files (HDLFS).

```
<dlf_path> ::= hdlfs://<dlf_endpoint>/<dlf_object_id>
```

**hdlsf_endpoint** Specifies the server address given from data lake Files. It consists of the data lake Files container id and the landscape.

**hdlsf_object_id** Specifies a path or file name within the data lake Files container. For example, `hdlsf://example-file-container.files.hdl.hc-XXX.XXX.hanacloud.ondemand.com/directory/sample.dat`

**hdlsf_path**

Specifies the location for the SAP HANA Cloud, Data Lake Files.

```plaintext
<hdlsf_path> ::= hdlfs://<hdlsf_endpoint>/<hdlsf_object_id>
```

**Prerequisites:**

- Data lake Files file container
- Client certificate and private key pair, registered in the data lake Files container and the root certificate
- Data lake Files SSL Certificate

**hdlsf_endpoint** Specifies the server address given from data lake Files. It consists of the data lake Files container id and the landscape.

**hdlsf_object_id** Specifies a path or file name within the data lake Files container. For example, `hdlsf://example-file-container.files.hdl.hc-XXX.XXX.hanacloud.ondemand.com/directory/sample.dat`

**path**

Specifies the path of the directory to be scanned for import data.

```plaintext
<path> ::= <string_literal>
```

**CREDENTIAL purpose_def**

Specifies the name of the credential defined in the CREATE CREDENTIAL statement. Since the credentials are defined within the credential, they no longer appear as plain text as part of import statements. The WITH CREDENTIAL clause cannot be specified when `<cloud_path>` contains credentials. The WITH CREDENTIAL clause is required for imports from SAP HANA Cloud, Data Lake Files, but is optional for all other cloud platforms.

**Permissions**

**Requires:**

- IMPORT system privilege
  - CREATE TEMPORARY TABLE privilege on the current schema

**Description**

The IMPORT SCAN statement searches the given path for objects exported with the EXPORT statement and stores its results in the following session-local temporary table at current_schema of the session if the
statement is executed successfully. If current_schema of the session is invalid, then IMPORT SCAN throws an exception as it cannot create its result table of the current_schema.

<current_schema>.#IMPORT_SCAN_RESULT

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>The database of the imported object</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>The schema of the object</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>The name of the object</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>The type of the object (TABLE, VIEW, and so on)</td>
</tr>
<tr>
<td>EXISTS</td>
<td>TINYINT</td>
<td>Does the object already exist in the system? (0/1)</td>
</tr>
</tbody>
</table>

Example

Scan the path 'data/import_path' for import data.

```
IMPORT SCAN 'data/import_path';
```

Select the data in the #IMPORT_SCAN_RESULT table to get the result of the scan.

```
SELECT * FROM #IMPORT_SCAN_RESULT;
```

Related Information

EXPORT Statement (Data Import Export) [page 997]

4.10.1.131 INSERT Statement (Data Manipulation)

Adds a record to a table.

Syntax

```
INSERT INTO <table_name> [ PARTITION <num> ]
[ [ AS ] <table_alias> ]
```
Syntax Elements

table_name

Specifies the table or view where the insert is to be performed, with optional schema name.

<table_name> ::= [ [ <database_name>.]<schema_name>.]<identifier>

For views, the view must only reference a single table, have no GROUP BY clause, and have an INSTEAD OF TRIGGER defined.

For linked database, <database_name> is the name of the remote source. <identifier> is the name of the table on the remote source.

PARTITION num

Specifies the partition number where the values are being inserted.

table_alias

Specifies an alias for the table that can be referenced later in the statement.

If you specify <table_alias> without using the AS keyword, then you must specify a <column_list_clause> that uses the <table_alias> in each column definition.

column_list_clause

Specifies a list of column identifiers, ordered in the order of values in the <value_list_clause> or <subquery>.

If the column list is omitted, then the database performs the insert using all the columns in the table. A column that is not included in the column list is filled using the column’s default value. When altering a table, always specify the column names to avoid potential issues where the order of the columns has changed.

value_list_clause

Specifies a list of values, or expressions evaluating to values, that are inserted into the table.

overriding_clause

Specifies a subquery. For more information about subqueries, see the SELECT statement.
Specifies the override behavior.

```sql
<overriding_clause> ::= OVERRIDING { SYSTEM | USER } VALUE
```

Use this clause with `<subquery>`, and only when the table being inserted into has an identity column. You can force the system to use the generated sequence value for identity column by specifying OVERRIDING USER VALUE, and to ignore the value provided from the SELECT.

OVERRIDING USER VALUE is required if the identity column was defined as GENERATED ALWAYS clause, and it is optional if the column was defined as GENERATED BY DEFAULT.

If OVERRIDING USER VALUE is not specified for a GENERATED BY DEFAULT identity column, the value provided for the column in the SELECT is inserted.

OVERRIDING SYSTEM VALUE tells the system to use the value from the select for an identity column.

```sql
<with_clause> ::= WITH <alias> AS ( <subquery> ) SELECT <expression> FROM <alias>
```

Defines the columns and the data to insert into them.

```sql
<with_clause> ::= WITH <alias> AS ( <subquery> ) SELECT <expression> FROM <alias>
```

For information on hints, see the HINT clause of the SELECT statement.

**Description**

The values to be inserted can either be values, expressions, or the result of a subquery. If the subquery used does not return any records, then the database does not insert any records into the table.

Always define the `<column_list_clause>`. This practice helps to protect your INSERT queries from damaging data if the target table schema is modified.

**Examples**

Create table T:

```sql
CREATE TABLE T (KEY INT PRIMARY KEY, VAL1 INT, VAL2 NVARCHAR(20));
```

Insert a row into table T.

```sql
INSERT INTO T VALUES (1, 1, 'The first');
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL1</th>
<th>VAL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>The first</td>
</tr>
</tbody>
</table>
Insert a new row into table T using column list to specify which columns should receive the input values.

```
INSERT INTO T (KEY, VAL2) VALUES (2,3);
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL1</th>
<th>VAL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>The first</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>NULL</td>
</tr>
</tbody>
</table>

Insert a row into table T using a subquery.

```
INSERT INTO T SELECT 3, 3, 'The third' FROM DUMMY;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL1</th>
<th>VAL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>The first</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>The third</td>
</tr>
</tbody>
</table>

Insert values into an array value construction by enumeration.

```
INSERT INTO T1 VALUES ( 1, ARRAY ( 1, 2, 3, 4 ) );
```

Insert values into an array value construction by query.

```
CREATE ROW TABLE T0 ( C1 INT )
INSERT INTO T0 VALUES ( 21 )
INSERT INTO T0 VALUES ( 22 )
INSERT INTO T0 VALUES ( 23 )
INSERT INTO T1 VALUES ( 2, ARRAY( SELECT C1 FROM T0 ) )
```

Insert a NULL value into an ARRAY.

```
INSERT INTO T1 (ID) VALUES (3);
```

Use an alias for the table you are inserting values into.

```
INSERT INTO CXML_LOG_CONTENT AS t0 (t0.IS_COMPRESSED, CONTENT, t0.CONTENT_KEY, t0.CONTENT_NEED_ENCR, t0.CONTENT_SIZE, ID) VALUES ('true', EMPTY_BLOB(), NULL, '0', '333', '1234');
```

In the example below, the latter two INSERT statements demonstration inserting data into a table using the WITH clause:

```
CREATE TABLE t1(a INT, b INT, c INT);
CREATE TABLE t2(a INT, b INT, c INT);
INSERT INTO t1 VALUES(1,12,13);
INSERT INTO t2 VALUES(2,120,130);
INSERT INTO t1 WITH alias AS (SELECT * FROM t1) SELECT * FROM alias;
INSERT INTO t1 WITH w1 AS (SELECT * FROM t2) SELECT * FROM w1;
```
Use the alias t0 for the CXML_LOG_CONTENT table.

```sql
INSERT INTO CXML_LOG_CONTENT AS t0 (t0.IS_COMPRESSED, CONTENT, t0.CONTENT_KEY, t0.CONTENT_NEED_ENCR, t0.CONTENT_SIZE, ID) VALUES ('true', EMPTY_BLOB(), NULL, '0', '333', '1234');
```

In the example below, the first statement inserts values into table T using the `<table_name.column_name>` column definition. The second statement inserts values into the table named ALIAS_T using the `<table_alias.column_name>` column definition.

```sql
INSERT INTO T (T.KEY, T.VAL2) VALUES(2,3);
INSERT INTO T ALIAS_T (ALIAS_T.KEY, ALIAS_T.VAL2) VALUES(2,3);
```

Related Information

Introduction to SQL [page 27]
Expressions [page 56]
Predicates [page 61]
SELECT Statement (Data Manipulation) [page 1104]

4.10.1.132  LOAD Statement (Data Manipulation)

Explicitly loads column store table data into memory instead of upon first access.

Syntax

```sql
LOAD <table_name> [HISTORY]
{ DELTA
  | ALL
  | {<column_name>, ...}
}
```

Syntax Elements

**table_name**

Specifies the name of the table to be loaded into memory, with optional schema name.

```sql
<table_name> ::= [<schema_name>.]<identifier>
<schema_name> ::= <unicode_name>
```

HISTORY
Use the HISTORY option when `<table_name>` is the name of a history table. The default operation of the LOAD command loads the current state of the given table into memory. If you explicitly require the table history to be loaded into memory, a separate LOAD statement with the HISTORY option enabled has to be executed. So to load a history completely into memory, two LOAD statements have to be executed, with and without HISTORY option enabled.

If you do not explicitly load table history into memory, then the SAP HANA database automatically loads this data upon first access.

**DELTA**

Specifies that the delta part of a column store table is loaded into memory. As the column store is read, optimized, and compressed, deltas are used to optimize insert or update operations.

**ALL**

Specifies that all current data of the column store table, including its delta, is loaded into memory. This option does not load the table history.

**column_name**

Specifies the name of the column to be loaded into memory.

```
<column_name> ::= <identifier>
```

**Description**

The LOAD statement explicitly loads column store table data into memory instead of upon first access.

A LOAD statement without HISTORY option only loads the current table data and the table delta into memory, but not the history-delta and history-main.

The use of this statement requires the UPDATE privilege.

**Examples**

Create column table A.

```
CREATE COLUMN TABLE A (A INT, B INT);
```

Load table A into memory.

```
LOAD A all;
```

Load the columns A and B of table A into memory.

```
LOAD A (A,B);
```

Query the load status of table A using the m_cs_tables monitoring view.

```
SELECT loaded FROM m_cs_tables WHERE table_name = 'A';
```
4.10.1.133 LOCK TABLE Statement (Transaction Management)

Acquires an exclusive lock for a table.

Syntax

```sql
LOCK TABLE <table_name>    
[ PARTITION <partition_id> ]  IN { EXCLUSIVE | INTENTIONAL EXCLUSIVE } MODE   
[ <wait_nowait> ]
```

Syntax Elements

- **table_name**
  Specifies the table to lock, with the optional schema name.

  ```
  <table_name> ::= [<schema_name>.]<identifier>  
  <schema_name> ::= <unicode_name>
  ```

- **partition_id**
  Specifies the logical partition ID of the partition to lock. Specifying the PARTITION clause allows for parallel table optimizations such as delta merge operations. Without the PARTITION clause, these optimizations are not possible.

  ```
  <partition_id> ::= <unsigned_integer>
  ```

- **EXCLUSIVE**
  Locks the table in EXCLUSIVE mode. EXCLUSIVE locks are acquired explicitly by LOCK TABLE, but are also acquired implicitly by DDL commands. EXCLUSIVE locks have the following characteristics:

  - The transaction that holds the lock can read and write to the table. All other transactions can read from the table but cannot write to it.
  - When a table has an EXCLUSIVE lock on it, all other lock requests by other transactions are blocked.
  - The database releases acquired locks at the end of the transaction.

- **INTENTIONAL EXCLUSIVE**
  Locks the table in INTENTIONAL EXCLUSIVE mode. However, INTENTIONAL EXCLUSIVE locks are only acquired by DML commands; use EXCLUSIVE mode instead. The description for INTENTIONAL
EXCLUSIVE locks (acquired implicitly by DML commands only) is being provided here to explain the behavioral differences between the two different locks that are used in the database.

- Multiple transactions can acquire an INTENTIONAL EXCLUSIVE lock.
- When a table has an INTENTIONAL EXCLUSIVE lock on it, incoming EXCLUSIVE lock requests for the table by other transactions are blocked, whereas other INTENTIONAL EXCLUSIVE locks are allowed.
- The database releases acquired locks at the end of the transaction.

### wait_nowait

Specifies when to return an error if the table lock could not be acquired.

```plaintext
<wait_nowait> ::= WAIT <unsigned_integer> | NOWAIT
```

If WAIT `<unsigned_integer>` is specified, then the statement waits up to the specified number of seconds for the lock. If the statement fails to acquire a lock after waiting, then it returns an error message indicating a timeout. If NOWAIT is specified, and the statement fails to acquire a table lock, then an error is returned indicating that the resource is busy. WAIT 0 is equivalent to NOWAIT.

When a timeout error is returned, the current transaction is not rolled back.

If `<wait_nowait>` is not specified, then the default behavior reflects the lock_wait_timeout transaction timeout setting located in `indexserver.ini`.

### Description

LOCK TABLE acquires a lock for a table.

If EXCLUSIVE is specified and there are views that reference the table, then the LOCK TABLE statement also attempts to lock the views in exclusive mode so that new transactions cannot hold locks on any dependent views. If all of the view locks cannot be acquired at that time, then the statement returns a non-zero SQLCODE even though the table itself is locked. Schema locks are only released upon the COMMIT or ROLLBACK of a transaction.

### Example

The following statements create a table `A`, and acquire an exclusive lock on it:

```sql
CREATE ROW TABLE A (A INT PRIMARY KEY, B INT);
LOCK TABLE A IN EXCLUSIVE MODE;
```

The following statement attempts to acquire an exclusive lock for table `A`, and specifies that an error should be returned if the lock cannot be immediately obtained.

```sql
LOCK TABLE A IN EXCLUSIVE MODE NOWAIT;
```

The following statements create a partitioned table `PART_A`, and acquire an exclusive lock on partition 1:

```sql
CREATE COLUMN TABLE PART_A (A INT) PARTITION BY HASH(A) PARTITIONS 3;
LOCK TABLE PART_A PARTITION (1) IN EXCLUSIVE MODE;
```
4.10.1.134 MERGE DELTA Statement (Data Manipulation)

Merges the column store table delta storage to the table's main storage.

Syntax

```
MERGE [ HISTORY ] DELTA OF <table_name> [ PART <n> ]
[ WITH PARAMETERS ( <parameter_list>, ... ) ]
[ FORCE REBUILD ]
```

Syntax Elements

**HISTORY**
Merges the history-delta storage of a table into the history-main storage for column store temporal tables.

**DELTA OF table_name**
Specifies the table where the delta merge occurs, with the optional schema name.

```
<table_name> ::= [ <schema_name>.]<identifier>
<schema_name> ::= <unicode_name>
```

**PART n**
Merges the delta of a specific table partition to the table's main storage.

**WITH PARAMETERS**
Specifies options that are specific to the column store.

```
<parameter_list> ::= <parameter> [, <parameter> [,... ] ]
<parameter> ::= <parameter_name> = <parameter_setting>
<parameter_name> ::= 'SMART_MERGE'
<parameter_setting> ::= 'ON' | 'OFF'
```

When SMART_MERGE is ON, the database does a smart merge based on merge criteria configured in indexserver.ini.

**FORCE REBUILD**
Force a delta merge even if the delta storage is empty and no deleted rows exist in the main storage that could be discarded. Use this option to create a new main storage matching the latest table definition (that is, reflecting current persistent memory preferences).
**Description**

As a column store table is read, optimized, and compressed, deltas are introduced to optimize insert or update operations. All insertions are passed to the delta storage. At a certain point in time the delta changes to a table can be merged into the table main storage.

**Permissions**

The UPDATE privilege on the column store table is required for performing a delta merge.

**Examples**

Create history column table TableA.

```sql
CREATE HISTORY COLUMN TABLE TableA(c NVARCHAR(1000)) PARTITION BY ROUNDROBIN PARTITIONS 2;
```

Merge the column store table delta storage to the tables main storage.

```sql
MERGE DELTA OF TableA;
```

Merge the column store table TableA using a smart merge.

```sql
MERGE DELTA OF TableA WITH PARAMETERS('SMART_MERGE' = 'ON');
```

Merge table TableA delta storage of partition 1 to the main storage of partition 1.

```sql
MERGE DELTA OF TableA PART 1;
```

Merge table TableA history-delta storage into its history-main storage.

```sql
MERGE DELTA OF TableA FORCE REBUILD;
```

Merge table TableA partition 1 history-delta storage to its history-main storage.

```sql
MERGE HISTORY DELTA OF TableA PART 1;
```
4.10.1.135 MERGE INTO Statement (Data Manipulation)

Merges data into an existing column store table.

Syntax

```sql
MERGE INTO <target_table> [ [ <partition_restriction> ] ] [ [ AS ] <alias> ]
USING <table_reference>
ON <search_condition> <merge_operation_specification>
```

Syntax Elements

- **<target_table>**
  - Specifies the target table (identifier) on which to perform the merge operation.

- **<partition_restriction>**
  - For a partitioned table, `<partition_restriction>` specifies the partition number in which the target rows are located.

  ```sql
  <partition_restriction> ::= PARTITION ( <partition_number> )
  <partition_number> ::= <unsigned_integer>
  ```

  If you do not specify a partition restriction, then the database checks all partitions for the merge. But if there is a partition restriction, then only the rows in the specified partition are merged. Use the TABLE_PARTITIONS system view to determine the partition number.

  The `<partition_restriction>` clause is supported for multistore tables but not extended store tables.

- **AS alias**
  - Specifies the alias to be used to refer to the table defined by `<target_table>`.

- **USING table_reference**
  - Specifies the source (identifier) of the data to be updated or inserted.

- **search_condition**
  - Specifies the criteria to use for matching values between `<table_reference>` and `<target_table>`.

- **merge_operation_specification**
  - Specifies merge operations to perform on `<table_reference>`.

  ```sql
  <merge_operation_specification> ::= { <when_matched_clause> | <when_not_matched_clause> | <when_matched_clause> <when_not_matched_clause> }
  ```

  `<when_matched_clause>` must precede `<when_not_matched_clause>` if you specify both; otherwise, an error is returned.

  - **when_matched_clause**
Specifies the update to perform when values match.

```sql
<when_matched_clause> ::=  WHEN MATCHED [ AND <search_condition> ] THEN <when_matched_specification>  <when_matched_specification> ::=  <update_specification>  | <delete_specification>  <update_specification> ::= UPDATE SET <set_clause_list>  <delete_specification> ::= DELETE
```

**when_not_matched_clause**

Specifies the insert to perform when values do not match.

```sql
<when_not_matched_clause> ::=  WHEN NOT MATCHED [ AND <search_condition> ] THEN  <when_not_matched_specification> ::=   INSERT [ ( <insert_column_list> ) ] VALUES <insert_value_list>  <insert_value_list> ::=  { <insert_value_element> [ , <insert_value_element> ] } ... }  <insert_value_element> ::=  <value_expression>  | <contextually_typed_value_specification>
```

**Description**

A MERGE INTO statement conditionally updates the rows of a table and/or inserts new rows into a table. The MERGE INTO statement differs from the REPLACE statement in the following ways:

- You must specify a data source.
- You can omit the update or the insert specifications.

**MULTIDRAG**

When you specify the WHEN MATCHED clause, a row must satisfy the join condition to be updated or deleted. If you specify the optional AND <search_condition> THEN ... clause, then a row must match both the join condition and the <search_condition> to be updated or deleted.

When you specify the WHEN NOT MATCHED clause, a row must not satisfy the join condition to be updated or deleted. If you specify the optional AND <search_condition> THEN INSERT ... clause, then a row must not satisfy the join condition but match the search condition to be inserted.

The existing SAP HANA <update_from> syntax can also be replaced by the MERGE INTO statement by omitting the MERGE WHEN NOT MATCHED clause.

On SAP HANA systems with dynamic tiering, the MERGE INTO statement is fully supported for multistore tables. It is not, however, supported in dynamic tiering extended store tables.

**Examples**

The following example shows a simple merge operation:

```sql
CREATE SCHEMA "my_schema";
```
DROP TABLE "my_schema".t1;
DROP TABLE "my_schema".t2;
CREATE ROW TABLE "my_schema".t1 (a INTEGER, b INTEGER);
INSERT INTO "my_schema".t1 VALUES(1,1);
CREATE ROW TABLE "my_schema".t2 (a INTEGER, b INTEGER);
INSERT INTO "my_schema".t2 VALUES(1,2);
INSERT INTO "my_schema".t2 VALUES(2,3);
MERGE INTO "my_schema".t1 USING "my_schema".t2 ON "my_schema".t1.a = "my_schema".t2.a
WHEN MATCHED THEN UPDATE SET "my_schema".t1.b = "my_schema".t2.b
WHEN NOT MATCHED THEN INSERT VALUES("my_schema".t2.a, "my_schema".t2.b);
SELECT * FROM "my_schema".t1;

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

The following merge example is equivalent to executing UPDATE t1 FROM t1,t2 SET b = t2.b WHERE t1.a = t2.a;

CREATE SCHEMA "my_schema";
DROP TABLE "my_schema".t1;
DROP TABLE "my_schema".t2;
CREATE ROW TABLE "my_schema".t1 (a INTEGER, b INTEGER);
INSERT INTO "my_schema".t1 VALUES(1,1);
CREATE ROW TABLE "my_schema".t2 (a INTEGER, b INTEGER);
INSERT INTO "my_schema".t2 VALUES(1,2);
INSERT INTO "my_schema".t2 VALUES(2,3);
MERGE INTO "my_schema".t1 USING "my_schema".t2 ON "my_schema".t1.a = "my_schema".t2.a
WHEN MATCHED THEN UPDATE SET "my_schema".t1.b = "my_schema".t2.b;
SELECT * FROM "my_schema".t1;

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The following example only inserts values:

CREATE SCHEMA "my_schema";
DROP TABLE "my_schema".t1;
DROP TABLE "my_schema".t2;
CREATE ROW TABLE "my_schema".t1 (a INTEGER, b INTEGER);
INSERT INTO "my_schema".t1 VALUES(1,1);
CREATE ROW TABLE "my_schema".t2 (a INTEGER, b INTEGER);
INSERT INTO "my_schema".t2 VALUES(1,2);
INSERT INTO "my_schema".t2 VALUES(2,3);
MERGE INTO "my_schema".t1 USING "my_schema".t2 ON "my_schema".t1.a = "my_schema".t2.a
WHEN NOT MATCHED THEN INSERT VALUES("my_schema".t2.a, "my_schema".t2.b);
SELECT * FROM "my_schema".t1;

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
The following example deletes matched rows and returns an empty result:

```sql
CREATE ROW TABLE T1 (A INTEGER, B INTEGER);
INSERT INTO T1 VALUES (1,0);
INSERT INTO T1 VALUES (2,0);
CREATE ROW TABLE T2 (A INTEGER, B INTEGER);
INSERT INTO T2 VALUES (1,1);
INSERT INTO T2 VALUES (2,2);
INSERT INTO T2 VALUES (3,3);
INSERT INTO T2 VALUES (4,4);
MERGE INTO T1 USING T2 ON T1.A = T2.A
WHEN MATCHED THEN DELETE;
SELECT * FROM T1;
```

The following example has search conditions applied and returns the table below:

```sql
CREATE COLUMN TABLE T1 (A INTEGER, B INTEGER);
INSERT INTO T1 VALUES (1,0);
INSERT INTO T1 VALUES (2,0);
CREATE COLUMN TABLE T2 (A INTEGER, B INTEGER);
INSERT INTO T2 VALUES (1,1);
INSERT INTO T2 VALUES (2,2);
INSERT INTO T2 VALUES (3,3);
INSERT INTO T2 VALUES (4,4);
MERGE INTO T1 USING T2 ON T1.A = T2.A
WHEN MATCHED AND T1.A > 1 THEN UPDATE SET B = T2.B
WHEN NOT MATCHED AND T2.A > 3 THEN INSERT VALUES (T2.A, T2.B);
SELECT * FROM T1;
```

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

The following example inserts data into specific partitions in tables T1 and T2 and then merges data matched data in table T2 into partition 1 on table T1:

```sql
CREATE COLUMN TABLE T1 (a INT, b INT) PARTITION BY RANGE(a) (PARTITION 0 <=
VALUES < 10, PARTITION OTHERS);
CREATE COLUMN TABLE T2 (a INT, b INT) PARTITION BY RANGE(a) (PARTITION 0 <=
VALUES < 10, PARTITION OTHERS);
INSERT INTO T1 PARTITION(1) VALUES(1, 1);
INSERT INTO T1 PARTITION(2) VALUES(10, 1);
INSERT INTO T2 PARTITION(2) VALUES(10, 1);
MERGE INTO T1 PARTITION(1) USING T2 ON T1.b = T2.b
WHEN MATCHED THEN UPDATE SET T1.b = 2
WHEN NOT MATCHED THEN INSERT VALUES (10, 1);
SELECT * FROM T1;
```

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>
4.10.1.136 RECOVER DATA Statement (Backup and Recovery)

Recovers the SAP HANA database using a specific data backup.

Syntax

```
RECOVER DATA [ FOR <database_name> ]
[ <system_source_definition> ]
<backup_definition_for_recovery>
CLEAR LOG
[ <additional_option_list> ]
[ <ignore_option> ]
```

**Syntax Elements**

**database_name**

Working from the system database, this option specifies the tenant database to be recovered.

```
<database_name> ::= FOR <identifier>
```

If no database name is specified, the local database is assumed.

**system_source_definition**

For third-party backup tools, identifies the source system that was used to create the backups.

This option can only be used in combination with USING BACKINT.

```
<system_source_definition> ::= USING SOURCE '<system_id>'
```

USING SOURCE specifies that the data backup is of a database with a different SID or name than the target system.

For example, if you are recovering the system database for system PRI1, the statement would be as follows:

```
<system_source_definition> ::= USING SOURCE 'SYSTEMDB@PRI1'
```

To recover a tenant database using a Backint-based backup that was created in a system identified by the system ID PRI1 and the name Tenant_1, the option would be:

```
<system_source_definition> ::= USING SOURCE 'Tenant_1@PRI1'
```
To recover a Backint-based data backup from a single container system identified by the system ID PR1, use the following option:

```
USING SOURCE 'PR1'
```

**backup_definition_for_recovery**

Specifies recovery settings for the backup.

```
<backup_definition_for_recovery> ::= 
    USING FILE ('<prefix>') 
    | USING BACKINT ('<prefix>') 
    | USING SNAPSHOT 
    | USING BACKUP_ID <backup_id> [ <catalog_data_path_definition> ]
<catalog_data_path_definition> ::= 
    USING CATALOG BACKINT 
    | USING CATALOG PATH ('<path>') 
    | USING DATA PATH ('<path_list>')
```

**USING SNAPSHOT** recovers the database from a data snapshot.

**Note**

You can recover SAP HANA using a data snapshot of an SAP HANA database with one or more tenant databases. When you recover SAP HANA from a data snapshot, you need to recover both the system database and each tenant database separately.

**USING BACKUP ID** specifies the ID of the full data backup to use for recovery.

```
<backup_definition_for_recovery> can point to either the file system or to a third-party backup tool.

**USING CATALOG BACKINT** specifies that the location of the backup catalog is a third-party backup tool.

**USING CATALOG PATH** specifies that the location of the backup catalog is in the file system.

An SQL statement can have either **USING CATALOG BACKINT** or **USING CATALOG PATH**, but not both.

To use a backup catalog that is in a third-party backup tool, use the option **USING CATALOG BACKINT**. When a backup catalog in a third-party backup tool is accessed, only the configured location is accessed. No traversal search for a backup catalog is done. You can specify a particular data backup using the **BACKUP_ID** option.

In the file system, SAP HANA searches the specified locations for the backup catalog in the following order:

<table>
<thead>
<tr>
<th>Path Specified...</th>
<th>Where Searched...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The path specified is searched for a backup catalog.</td>
</tr>
<tr>
<td>In the SQL statement (optional):</td>
<td>No subdirectories are searched.</td>
</tr>
<tr>
<td><strong>USING CATALOG PATH</strong></td>
<td></td>
</tr>
</tbody>
</table>
### Path Specified...Where Searched...

2. Configured path.

The path to be searched is specified by the following parameter:

- `basepath_catalogbackup`

This parameter is in the persistence section of the global.ini parameter file.

**USING DATA PATH** is an additional option for **USING CATALOG PATH.** **USING DATA PATH** specifies the location of the data backup in the file system.

In the file system, SAP HANA searches the specified locations for backups in the following order:

<table>
<thead>
<tr>
<th>Path Specified...</th>
<th>Where Searched...</th>
</tr>
</thead>
</table>
| 1. **In the SQL statement (optional):**
  **USING DATA PATH** `<path list>` | The specified paths are searched for data backups or log backups. All subdirectories are searched. If a specified path is **relative,** the path is based on the backup paths specified by the following parameters: `basepath_databackup`
  This parameter is in the persistence section of the global.ini parameter file. |
| 2. Configured paths. | The paths to be searched are specified by the following parameters: `basepath_databackup`
  This parameter is in the persistence section of the global.ini parameter file. |
| 3. If no backups are found in the paths for 1. or 2. | The paths specified in the original backup catalog are searched for the backups. |

### CLEAR LOG

The CLEAR **LOG** option **must be used** here to prevent entries from the log area from being used for the recovery. No log entries from the log area are replayed, and the log area is initialized. As a consequence, the content of the log area is lost.

**additional_option_list**

Additional options for recovery.

```plaintext
<additional_option_list> ::= <additional_option> [...]  

<additional_option> ::= SET LICENSE '<license>' | TOOLOPTIO  
  'n' | INCLUDE CONFIGURATION
```

**SET LICENSE**
SET LICENSE allows you to set the license during the recovery.

TOOLOPTIOIN

TOOLOPTIOIN is only available for BACKINT.

TOOLOPTIOIN passes a vendor-specific parameter string to a third-party backup tool. The string has a maximum length of 4096 bytes.

COMPRESSED (BACKUP DATA option)

To recover compressed backups, no additional action is required, as RECOVER DATA automatically recognizes if a backup is compressed.

INCLUDE CONFIGURATION

During the recovery, INCLUDE CONFIGURATION restores the customer-specific configuration files (.ini files that have been changed from the default) from the metadata of the backup.

The customer-specific configuration files are available in a data backup if they were explicitly included when the data backup was created.

For more information, see BACKUP DATA Statement (Backup and Recovery).

i Note

If the backup includes parameters that require a database restart to take effect, you need to restart the database after the recovery.

To check which parameters require a database restart, use the system view CONFIGURATION_PARAMETER_PROPERTIES. Column RESTART_REQUIRED indicates whether a database restart is required by a particular parameter.

For more information, see CONFIGURATION_PARAMETER_PROPERTIES System View.

If the backed-up configuration parameters need to take effect before the recovery starts, you need to manually restore the configuration.

For more information, see Check Individual Backups Inside an SAP HANA Installation (hdbbackupdiag, hdbbackupcheck) (hdbbackupcheck option --dump -c <.ini file>).

During a recovery, no .ini files are deleted. A copy of the current configuration files is archived in a .tgz file that is stored in the same directory as the backed-up configuration files. The .tgz file name contains a timestamp. For example, inifiles_20211006-092603.tgz. If you have access to the file system, you can manually restore the .ini files, if required.

No redo logging is done for the user configuration files. During a recovery, the configuration state recorded in the backup is used. It is not possible to recover the user configuration to a specific point in time.

During a recovery, .ini files are processed as follows:

<table>
<thead>
<tr>
<th>If...</th>
<th>What is Recovered?</th>
</tr>
</thead>
<tbody>
<tr>
<td>An .ini file exists on the instance to be recovered, but is not in the data backup</td>
<td>The .ini file remains unchanged.</td>
</tr>
<tr>
<td>An .ini file exists in the data backup, and also already exists on the instance to be recovered</td>
<td>That .ini file is overwritten.</td>
</tr>
</tbody>
</table>
If...

What is Recovered?

An `.ini` file exists in the data backup, but not on the instance to be recovered

That `.ini` file is restored from the data backup.

---

### ignore_option_list

```
<ignore_option_list> ::= <ignore_option> [...]
```

```
<ignore_option> ::= IGNORE WORKERGROUPS | IGNORE COMPATIBILITY CHECK
```

### IGNORE WORKERGROUPS

By default, when you recover a database from an indexserver that has worker groups, the target system must have worker groups with the same names as in the source system. Otherwise, the recovery fails.

**IGNORE WORKERGROUPS** removes this restriction. If worker groups with matching names are found on the target system, they are used. If no worker groups with matching names are found, the worker groups of the backup service are changed to the default to match those of the target system.

### IGNORE COMPATIBILITY CHECK

By default, the version of the backup is checked for compatibility with the SAP HANA revision. You have the option to recover SAP HANA without performing this check.

⚠️ **Caution**

Disabling the compatibility check may have unforeseen consequences. Use this option only in exceptional situations.

For more information, see SAP Note 1948334 (SAP HANA Database Update Paths for SAP HANA Maintenance Revisions)

---

### Description

The database is recovered to the state of the full backup.

Incremental, differential, and log backups are ignored. No log entries from the log area are replayed, and the log area is initialized. As a consequence, the content of the log area is lost.

After the recovery, the SAP HANA database is online and can be used by applications.

---

### Permissions

Privilege needed:

```
FOR <database_name>
```
Examples

Scenario 1: Recover Without Replaying the Redo Log
In this example, the data backup is taken from the specified directory `/backup`, and has the backup prefix THURSDAY. No log entries from the log area are replayed, and the log area is initialized.

```
RECOVER DATA USING FILE ('/backup/THURSDAY') CLEAR LOG;
```

Scenario 2: Copy a Tenant Database from a Third-Party Backup Tool
In this example, a source tenant database named Tenant_0 is copied using a Backint-based data backup to a target tenant database named Tenant_1. The Backint-based data backup is stored in a third-party backup tool that is identified by the source system ID (SID) (in this example, PR1) and the tenant database name Tenant_0. The Backint-based data backup name is specified by a backup prefix (in this example, the timestamp for Thursday, November 26, 2020 at 9 a.m.).

```
RECOVER DATA FOR Tenant_1 USING SOURCE 'Tenant_0@PR1' USING BACKINT('2020-11-26') CLEAR LOG;
```

Scenario 3: Recover Using a Backup in the File System and a Backup Catalog in a Third-Party Backup Tool
In this example, a tenant database PR2 in SAP HANA system PR2 is recovered using a data backup of tenant database PR1 in SAP HANA system PR1. The data backup of PR1 has BACKUP_ID 1591709192198. The backup catalog is in a third-party backup tool. The data backup was created in the file system of the source system, and was copied to the file system of the target system. USING DATA PATH is used to override the path specified in the source backup catalog with the file system directory that currently contains the data backup.

```
RECOVER DATA FOR PR2 USING SOURCE 'PR1@PR1' USING BACKUP_ID 1591709192198 USING DATA PATH ('/hana/PR1/backup/data') CLEAR LOG
```

Scenario 4: Recover Using a Backup in a Third-Party Backup Tool and a Backup Catalog in the File System
In this example, a tenant database PR2 in SAP HANA system PR2 is recovered using a data backup of tenant database PR1 in SAP HANA system PR1. The data backup of PR1 is in a third-party backup tool and has BACKUP_ID 1591709192198. The backup catalog is in the file system of the target system. For this reason, USING CATALOG PATH is used to specify the location of the backup catalog to be used.

```
RECOVER DATA FOR PR2 USING SOURCE 'PR1@PR1' USING BACKUP_ID 1591709192198 USING CATALOG PATH ('/hana/PR1/backup/catalog') CLEAR LOG
```

Related Information

SAP HANA Recovery
Include User Configuration in Backup or Recovery
Check Individual Backups Inside an SAP HANA Installation (hdbbackupdiag, hdbbackupcheck)
BACKUP DATA Statement (Backup and Recovery) [page 699]
RECOVER DATABASE Statement (Backup and Recovery) [page 1070]
4.10.1.137  RECOVER DATABASE Statement (Backup and Recovery)

Recovers an SAP HANA database to a specific point in time or to a specific log position.

Syntax

```
RECOVER DATABASE [ FOR <database_name> ]
   { <until_definition> 
     [ <system_source_definition> ]
     [ <strategy_using_definition> ]
     [ <check_access_definition> ]
     [ <ignore_option_list> ]
   }
```

Syntax Elements

- **database_name**
  - This option is only available for SAP HANA tenant databases.
  - Recovers a tenant database to a specific point in time.

  `<database_name> ::= <identifier>`

- **until_definition**
  - Defines the target of the recovery.

  `<until_definition> ::= 
    UNTIL LOG POSITION <log_pos> AT VOLUME <volume_id> [ <recover_additional_option_list> ]
    | UNTIL TIMESTAMP '<timestamp>' [ <recover_additional_option_list> ]`

  `<timestamp>` is interpreted as UTC time. For example, 3:00 p.m. means 3:00 p.m. UTC, not 3:00 p.m. in your local time zone.

  **recover_additional_option_list**
  - Specifies additional recovery options.
CLEAR LOG
CLEAR LOG initializes the log area. The content of the log area is permanently lost. No log entries from the log area are used for the recovery.

If they are needed, the log entries from the log backups are replayed.

You must specify CLEAR LOG in the following situations:

- The log area is unusable.
- You are recovering the database to a different system.

SET LICENSE

SET LICENSE allows you to specify a license key during the recovery.

**Note**
To recover compressed backups, no additional action is required, as RECOVER DATABASE automatically recognizes if a backup is compressed.

INCLUDE CONFIGURATION

INCLUDE CONFIGURATION restores the customer-specific configuration files (.ini files that have been changed from the default) during the recovery.

The customer-specific configuration files are available in a data backup if they were explicitly included when the data backup was created.

For more information, see BACKUP DATA Statement (Backup and Recovery).

**Note**
If the backup includes parameters that require a database restart to take effect, you need to restart the database after the recovery.

To check which parameters require a database restart, use the system view CONFIGURATION_PARAMETER_PROPERTIES. Column RESTART_REQUIRED indicates whether a database restart is required by a particular parameter.

For more information, see CONFIGURATION_PARAMETER_PROPERTIES System View.

If the backed-up configuration parameters need to take effect before the recovery starts, you need to manually restore the customer-specific configuration.

For more information, see Check Individual Backups Inside an SAP HANA Installation (hdbbackupdiag, hdbbackupcheck) (hdbbackupcheck option --dump -c <.ini file>).

During a recovery, no .ini files are deleted. A copy of the current configuration files is archived in a .tgz file that is stored in the same directory as the backed-up configuration files. The .tgz file name contains a timestamp. For example, inifiles_20211006-092603.tgz. If you have access to the file system, you can manually restore the .ini files, if required.
No redo logging is done for the user configuration files. During a recovery, the configuration state recorded in the backup is used. It is not possible to recover the user configuration to a specific point in time.

During a recovery, `.ini` files are processed as follows:

<table>
<thead>
<tr>
<th>If...</th>
<th>What is Recovered?</th>
</tr>
</thead>
<tbody>
<tr>
<td>An <code>.ini</code> file exists on the instance to be recovered, but is not in the data backup</td>
<td>The <code>.ini</code> file remains unchanged.</td>
</tr>
<tr>
<td>An <code>.ini</code> file exists in the data backup, and also already exists on the instance to be recovered</td>
<td>That <code>.ini</code> file is overwritten.</td>
</tr>
<tr>
<td>An <code>.ini</code> file exists in the data backup, but not on the instance to be recovered</td>
<td>That <code>.ini</code> file is restored from the data backup.</td>
</tr>
</tbody>
</table>

**system_source_definition**

Identifies the source system.

This option can only be used if a data backup from a third-party backup tool is used for the recovery.

```plaintext
<system_source_definition> ::= USING SOURCE '<system_id>'
```

**USING SOURCE** specifies that the data backup is of a database with a different SID or name than the target system.

To recover a tenant database using a Backint-based backup that was created in a system identified by the system ID PR1 and the name `TENANT_1`, the option would be:

```plaintext
<system_source_definition> ::= USING SOURCE 'TENANT_1@PR1'
```

To recover a Backint-based data backup from a single container system identified by the system ID PR1, use the following option:

```plaintext
USING SOURCE 'PR1'
```

**strategy_using_definition**

Specifies the locations to search for the backup catalog, the data backup, and the log backups.

```plaintext
<strategy_using_definition> ::= <path_definition> [ <start_from_definition> ]
```

```plaintext
<path_definition> ::= [ <catalog_path_definition> ] [ USING LOG PATH ( <path_list> ) ] [ USING DATA PATH ( <path_list> ) ]
```

**USING LOG PATH** specifies the location of the log backup in the file system.

**USING DATA PATH** specifies the location of the data backup in the file system.
In the file system, SAP HANA searches the specified locations for backups in the following order:

<table>
<thead>
<tr>
<th>Path Specified...</th>
<th>Where Searched...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong></td>
<td></td>
</tr>
<tr>
<td>In the SQL statement (optional):</td>
<td></td>
</tr>
<tr>
<td>USING LOG PATH <code>&lt;path list&gt;</code></td>
<td>The specified paths are searched for data backups or log backups.</td>
</tr>
<tr>
<td>USING DATA PATH <code>&lt;path list&gt;</code></td>
<td>All subdirectories are searched.</td>
</tr>
<tr>
<td></td>
<td>If a specified path is relative, the path is based on the backup paths specified by the following parameters:</td>
</tr>
<tr>
<td></td>
<td>basepath_databackup</td>
</tr>
<tr>
<td></td>
<td>basepath_logbackup</td>
</tr>
<tr>
<td></td>
<td>These parameters are in the persistence section of the <code>global.ini</code> parameter file.</td>
</tr>
<tr>
<td><strong>2.</strong></td>
<td></td>
</tr>
<tr>
<td>Configured paths.</td>
<td>The paths to be searched are specified by the following parameters:</td>
</tr>
<tr>
<td></td>
<td>basepath_databackup</td>
</tr>
<tr>
<td></td>
<td>basepath_logbackup</td>
</tr>
<tr>
<td></td>
<td>These parameters are in the persistence section of the <code>global.ini</code> parameter file.</td>
</tr>
<tr>
<td><strong>3.</strong></td>
<td></td>
</tr>
<tr>
<td>If no backups are found in the paths for 1. or 2.</td>
<td>The paths specified in the original backup catalog are searched for the backups.</td>
</tr>
</tbody>
</table>

```plaintext
<catalog_path_definition> ::= USING CATALOG BACKINT | USING CATALOG PATH ( '<path>' )
```

To use a backup catalog that is in a third-party backup tool, use the option `USING CATALOG BACKINT`. When a backup catalog in a third-party backup tool is accessed, only the configured location is accessed. No traversal search for a backup catalog is done. You can specify a particular data backup using the `BACKUP_ID` option.

In the file system, SAP HANA searches the specified locations for the backup catalog in the following order:

<table>
<thead>
<tr>
<th>Path Specified...</th>
<th>Where Searched...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong></td>
<td></td>
</tr>
<tr>
<td>In the SQL statement (optional):</td>
<td></td>
</tr>
<tr>
<td>USING CATALOG PATH</td>
<td>The path specified is searched for a backup catalog.</td>
</tr>
<tr>
<td></td>
<td>No subdirectories are searched.</td>
</tr>
</tbody>
</table>
2. Configured path.

The path to be searched is specified by the following parameter:

`basepath_catalogbackup`

This parameter is in the `persistence` section of the `global.ini` parameter file.

---

**path_list**

Defines the path to the backup.

```
<path_list> ::=   '<path>'   | <path_list>, '<path>'
```

**start_from_and_guid_definition**

```
<start_from_and_guid_definition> ::=   [<start_from_definition>] [guid_definition]
```

**start_from_definition**

```
<start_from_definition> ::=   USING BACKUP_ID <int_const>  | USING SNAPSHOT  | USING RESUME
```

**Note**

You can recover SAP HANA using a data snapshot of an SAP HANA database with one or more tenant databases. When you recover SAP HANA from a data snapshot, you need to recover both the system database and then each tenant database separately. The tenant databases cannot be recovered together in one single operation.

**i Note**

Using `RESUME` continues a canceled or aborted recovery, without repeating the whole recovery from the start.

A resumed recovery must use the **same backup catalog** as the recovery that was canceled or aborted. To specify the backup catalog, use the following option:

- File-based backups:
  - `USING CATALOG PATH ('<path>')`
- Third-party backup tools:
  - `USING CATALOG BACKINT`

**guid_definition**

Defines the starting point of the backup using the global unique ID (GUID) of a log history.
If \texttt{RECOVER DATA} was already executed, you may need to use a different log history than the one proposed by default. To use a specific log history, use \texttt{USING GUID}. You can find a GUID for a specific log history in the \texttt{backup.log}.

\[<\text{guid_definition}> ::= \text{USING GUID_ID 'guid>'}\]

\textbf{check_access_definition}

Checks whether the backups are available in accordance with the recovery strategy.

\texttt{CHECK ACCESS} checks whether the required backups are available before the recovery is started. If one or more backups are not available, you are notified of this before the recovery starts, and the recovery is canceled before any changes are made to the database. If you do not check the availability of the required backups before recovery starts, missing backups will only be detected during the recovery. In this situation, you may need to restart or resume the recovery.

\[<\text{check_access_definition}> ::= \text{CHECK ACCESS ALL} \mid \text{CHECK ACCESS USING FILE} \mid \text{CHECK ACCESS USING BACKINT}\]

\texttt{CHECK ACCESS USING FILE} checks for backups in the file system.

\texttt{CHECK ACCESS USING BACKINT} checks for backups created with a third-party backup tool.

\texttt{CHECK ALL} checks for backups in both the file system and the third-party backup tool.

\textbf{ignore_option_list}

\[<\text{ignore_option_list}> ::= <\text{ignore_option}> \ldots\]

\[<\text{ignore_option}> ::= \text{IGNORE DELTA DATA BACKUPS} \mid \text{IGNORE WORKERGROUPS} \mid \text{IGNORE COMPATIBILITY CHECK}\]

\texttt{IGNORE DELTA DATA BACKUPS} uses log backups, without using incremental and differential backups.

\texttt{IGNORE WORKERGROUPS}: By default, when you recover a database from an indexserver that has worker groups, the target system must have worker groups with the same names as in the source system. Otherwise the recovery fails. \texttt{IGNORE WORKERGROUPS} removes this restriction. If worker groups with matching names are found in the target system, those worker groups are used. If no worker groups with matching names are found, the worker groups of the backup service are changed to the default to match those of the target system.

\texttt{IGNORE COMPATIBILITY CHECK}: By default, the version of the backup is checked for compatibility with the SAP HANA revision. You have the option to recover SAP HANA without performing this check.

\[\text{Caution}\]

Disabling the compatibility check may have unforeseen consequences. Use this option only in exceptional situations.

For more information, see \textit{SAP Note 1948334 (SAP HANA Database Update Paths for SAP HANA Maintenance Revisions)}
**Description**

RECOVER DATABASE recovers an SAP HANA database to a specific point in time or to a specific log position.

To recover a database, it must be offline. However, you can still execute SQL statements on the online system database to recover an offline tenant database.

To recover the system database from the command line (without using SAP HANA cockpit), use the recoverSys.py tool. To recover a tenant database from the command line, use an SQL interface such as hdbsql.

For more information about backup and recovery, including information about using recoverSys.py, refer to the SAP HANA Administration Guide (SAP HANA Database Backup and Recovery).

**Permissions**

Privilege needed:

FOR <database_name>: DATABASE RECOVERY OPERATOR or DATABASE ADMIN

**Examples**

**Scenario 1: Recovery to a Point in Time**

The last complete data backup was created on Wednesday, October 21, 2020 at 8 a.m. CEST (6 a.m. UTC).

The database ran until 6pm CEST (4 p.m. UTC). You want to recover the database to its state at 5pm (3 p.m. UTC).

```
RECOVER DATABASE FOR TENANT_1 UNTIL TIMESTAMP '2020-10-21 15:00:00' CHECK ACCESS ALL;
```

The database timestamp is interpreted as UTC time. 3 p.m. therefore means 3 p.m. UTC and not 3 p.m. CEST.

Use the following statement to recover the tenant database named TENANT_1 to the specified point in time:

The specified timestamp is not included in the transactions to be redone. Only transactions with a commit timestamp before the target timestamp are redone.

**Scenario 2: Recovery to a Point in Time Using a Specific Directory**

Recover the tenant database TENANT_1 to a point in time with a specific timestamp, by using a dedicated directory for data backups and further directories containing log backups.

Use the following statement:

```
RECOVER DATABASE FOR TENANT_1 UNTIL TIMESTAMP '2020-10-21 15:00:00' USING DATA PATH ('/backup/WEDNESDAY/') USING LOG PATH ('/backup/logs1/','/backup/logs2/');
```
The backup catalog is taken from the configured default location, which could be a third-party backup tool or a directory in the file system.

For more information, see the SAP HANA Administration Guide (SAP HANA Database Backup and Recovery).

**Scenario 3: Recovery Using a Data Snapshot**

Use a data snapshot to recover an SAP HANA with multiple tenant databases to its most recent state. Based on the restored data snapshot, log entries from the log area and log backups are applied.

You need to recover the system database and then each tenant database separately.

To recover the system database to its most recent state, use the following statement:

```sql
RECOVER DATABASE UNTIL TIMESTAMP '2020-10-21 15:00:00' USING SNAPSHOT;
```

To recover a tenant database to its most recent state using the data snapshot, use the following statement:

```sql
RECOVER DATABASE FOR TENANT_1 UNTIL TIMESTAMP '2027-10-21 15:00:00' USING SNAPSHOT;
```

With USING SNAPSHOT, no paths are specified. Before the recovery is started, the data snapshot needs to be made available in the data area.

---

**Scenario 4: Recovery Using a Catalog Path**

Use the following statement to recover the tenant database named `TENANT_1` to recover the tenant database `TENANT_1` to a point in time with a specific timestamp, using different locations in the file system for log backups and the backup catalog.

```sql
RECOVER DATABASE FOR TENANT_1 UNTIL TIMESTAMP '2020-10-21' USING CATALOG PATH ('/hana/PR1/backup/catalog') USING LOG PATH ('/hana/HBB/backup/log') CHECK ACCESS ALL;
```

**Scenario 5: Recover Using a Specific Log History**

Recover the tenant database `TENANT_1` to a specified log position in a log history specified by its persistence global unique ID (GUID).

```sql
RECOVER DATABASE FOR TENANT_1 UNTIL LOG POSITION 7553856 at volume 2 CLEAR LOG USING GUID '41D53842-FA51-0017-3527-F4A470000005';
```

**Scenario 6: Resume a Recovery**

Step 1: Recover SAP HANA database named `PR1`.

```sql
RECOVER DATABASE FOR PR1
UNTIL TIMESTAMP '2020-10-21 05:00:00'
CLEAR LOG
USING CATALOG BACKINT
USING LOG PATH ('/usr/sap/PR1/HDB/backup/log/DB_PR1')
USING DATA PATH ('/usr/sap/PR1/SYS/global/hdb/backint/DB_PR1/')
USING BACKUP_ID 1540499452637
```
Step 2: After the recovery is canceled, resume the recovery.

```
RECOVER DATABASE FOR PR1
    UNTIL TIMESTAMP '2020-10-21 05:00:00'
    CLEAR LOG
    USING CATALOG BACKINT
    USING RESUME
    CHECK ACCESS USING FILE
```

**Scenario 7: Recover Using Backups in the File System and a Backup Catalog in a Third-Party Backup Tool**

In this example, a tenant database **PR2** in SAP HANA system **PR2** is recovered using backups of tenant database **PR1** in SAP HANA system **PR1**. The backup catalog is in a third-party backup tool. The data backup was created in the file system of the source system, and was copied to the file system of the target system. **USING DATA PATH** is used to override the path specified in the source backup catalog with the file system directory that currently contains the data backup.

```
RECOVER DATABASE FOR PR2 UNTIL TIMESTAMP '2020' CLEAR LOG USING SOURCE 'PR1@PR1'
    USING DATA PATH ('/hana/PR1/backup/data')
```

In this scenario, the log backups are in the third-party backup tool. If the log backups were in the file system, **USING LOG PATH** would be used to override the path specified in the source backup catalog with the file system directory that contains the log backup.

**i Note**

To find out which backups are required to complete a recovery, you can use the `hdbbackupdiag` tool. You can also use `hdbbackupdiag` to check whether these backups are available and can be accessed.

For more information, see [Checking Whether a Recovery is Possible](SAP HANA Administration Guide (SAP HANA Database Backup and Recovery)).

**Scenario 8: Recover Using Backups in a Third-Party Backup Tool and a Backup Catalog in the File System**

In this example, a tenant database **PR2** in SAP HANA system **PR2** is recovered using backups of tenant database **PR1** in SAP HANA system **PR1**. The backups are in the third-party backup tool. The backup catalog was created in the file system of the source system, and was copied to the file system of the target system. For this reason, **USING CATALOG PATH** is used to specify the location of the backup catalog to be used.

```
RECOVER DATABASE FOR PR2 UNTIL TIMESTAMP '2020' CLEAR LOG USING SOURCE 'PR1@PR1'
    USING CATALOG PATH ('/hana/PR1/backup/catalog')
```

**Related Information**

[SAP HANA Recovery](SAP HANA SQL Reference Guide for SAP HANA Platform)
Include User Configuration in Backup or Recovery
Change the Default Destination for File-Based Data Backups
Change the Log Backup Destination
Check Individual Backups Inside an SAP HANA Installation (hdbbackupdiag, hdbbackupcheck)
Recover SAP HANA From a Data Snapshot
BACKUP DATA Statement (Backup and Recovery) [page 699]
RECOVER DATA Statement (Backup and Recovery) [page 1064]
CONFIGURATION_PARAMETER_PROPERTIES System View [page 1521]
SAP Note 1948334
SAP Note 2050606

4.10.1.138  RECOVER ENCRYPTION ROOT KEYS Statement (Backup and Recovery)

This statement allows you to recover the encryption root keys or all LSS databases for a tenant database. To recover the encryption root keys or all LSS databases for a system database, use the hdbnsutil tool.

Syntax

```
RECOVER ENCRYPTION ROOT KEYS [ AND SETTINGS | (<root_keytype_list>) ] FOR <database_name> USING FILE <root_key_backup_definition_file> PASSWORD <password> [ SECURE STORE SECOND ACCESS PRIVATE KEY '<pemfile-content>' PASSWORD '<pemfile-password>' ]
```

Syntax elements

AND SETTINGS

AND SETTINGS recovers the full LSS backup. This includes keys, key management configuration, encryption control, and encryption settings.

⚠️ Caution

AND SETTINGS replaces all LSS databases.

If AND SETTINGS is not specified, only the encryption root keys are recovered. The encryption root keys from the backup are added, but not replaced.

The AND SETTINGS works only when the input file format is LSS backup and the secure store is also LSS. In all other cases, the statement will fail if the clause is used.

```
root_keytype_list ::= root_keytype [, root_keytype_list ]
```

Specifies the root key type.
If a `<root_keytype_list>` is not specified, all encryption root keys types that are stored in the specified file are recovered.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
</table>
| You can only specify `<root_keytype_list>` if the secure store is SSFS.

**database_name**

Specifies the name of the tenant database.

**root_keytype ::= PERSISTENCE | APPLICATION | BACKUP | LOG**

`<root_keytype>` can be PERSISTENCE, APPLICATION, BACKUP, or LOG.

**root_key_backup_definition_file ::= FILE (prefix | path, prefix)**

Specifies the encryption root key backup definition file in the file system.

The name of the backup file is constructed from the following elements:

`<prefix>_<database_name>_rootkeys`

If `<path>` is not specified, SAP HANA attempts to retrieve the backup location from the configuration parameter `<basepath_rootkeybackup>` in the `<persistence>` section of the `global.ini` file.

If the `<basepath_rootkeybackup>` parameter is not configured, the default backup path `$ {DIR_INSTANCE}/backup/sec` is used.

**password**

Specifies the root keys backup password that was used while the backup was created.

Using `ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD`, the password is set early during the database installation, and occasionally updated later. The currently set password is used when a secure store backup is created.

For more information, see `BACKUP ENCRYPTION ROOT KEYS Statement (Backup and Recovery)`.

**SECURE STORE SECOND ACCESS PRIVATE KEY `pemfile-content` PASSWORD `pemfile-password`**

`pemfile-content` Specifies the private key in .pem file format.

`pemfile-password` Specifies the password to decrypt the ENCRYPTED PRIVATE KEY

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>The parameter PRIVATE KEY <code>&lt;pemfile-content&gt;</code> must contain the content of a .pem file with the private key.</td>
</tr>
<tr>
<td>the private key can either be protected/encrypted or unprotected, where PASSWORD <code>&lt;pemfile-password&gt;</code> is used to unprotect/access/decrypt the private key:</td>
</tr>
<tr>
<td>• Protected/encrypted</td>
</tr>
<tr>
<td>• BEGIN ENCRYPTED PRIVATE KEY</td>
</tr>
<tr>
<td>• Unprotected</td>
</tr>
<tr>
<td>• BEGIN PRIVATE KEY</td>
</tr>
</tbody>
</table>

Secure store second access is only needed if you want to recover a backup that was created while LSS was configured with active external key management configuration that can no longer be reached.

With the second access key, the LSS stops using the KMS protection and reverts to local protection. Consequently, if you require KMS protection, you will need to set it up again.
Description

! Restriction
This statement can only be used to recover encryption root key backup or recover full LSS backup (all databases) from the file system.
Use with third-party backup tools is not supported.

The statement must be executed in the system database.
To prevent inconsistencies in the persistence layer, the tenant database must be offline.
It does not matter if the system database user (<sid>adm) is the owner of the backup file or not.

With LSS, RECOVER ENCRYPTION ROOT KEYS supports second access keys. If the connection to the key management system is lost, or the key was deleted, backups can be recovered using the second access key.

! Note
Using this statement, the encryption root keys or full LSS backup (all databases) can only be recovered for tenant databases. To recover the encryption root keys or full LSS backup (all databases) for a system database, use the hdbnsutil tool.
For more information about the hdbnsutil utility, see the SAP HANA Administration Guide.

Permissions
Privilege needed:
FOR <database_name>: DATABASE RECOVERY OPERATOR or DATABASE ADMIN

Example

RECOVER ENCRYPTION ROOT KEYS PERSISTENCE FOR testDB USING FILE ('/backup/HDB/rootkeys/backup001') PASSWORD <PASSWORD>

Recovers the encryption root keys for the root key type PERSISTENCE for the testDB database from the backup file /backup/Hs/data/backup001_testDB_rootkeys. Only works with SSFS.

RECOVER ENCRYPTION ROOT KEYS FOR testDB USING FILE ('/backup/HDB/rootkeys/backup001') PASSWORD MyBackupPwd123 SECURE STORE SECOND ACCESS PRIVATE KEY
Recover the encryption root keys from an LSS backup that is password-protected, and makes use of an explicitly given second access key which is also password-protected.

```
RECOVER ENCRYPTION ROOT KEYS FOR testDB USING FILE ('/backup/HDB/rootkeys/backup001') PASSWORD MyBackupPwd123 SECURE STORE SECOND ACCESS PRIVATE KEY '-----BEGIN PRIVATE KEY-----
MIIFH...
-----END PRIVATE KEY-----' PASSWORD ''
```

Recover the encryption root keys from an LSS backup that is password-protected, and makes use of an explicitly given second access key which is not password-protected.

**Related Information**

- BACKUP ENCRYPTION ROOT KEYS Statement (Backup and Recovery) [page 709]
- ALTER SYSTEM SET ENCRYPTION ROOT KEYS BACKUP PASSWORD Statement (System Management) [page 568]
- ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
- Import Backed-Up Root Keys or LSS Backup Before Database Recovery

### 4.10.1.139 REFRESH STATISTICS Statement (Data Definition)

Specifies a column that is part of the data sources.

**Syntax**

```
REFRESH STATISTICS { <data_statistics_name>[,,<data_statistics_name>[,...] ]
| ON <data_sources> [ [HAVING] <match_properties> ] }
```

**Syntax Element**

- **data_statistics_name**

  Specifies the name of the data statistics object.

  ```
  <data_statistics_name> ::= [<schema_name>..<identifier>]
  <schema_name> ::= <identifier>
  ```

- **data_sources**
Specifies the data source(s) of the data statistics objects.

```xml
<data_sources> ::=  
<table_name> [ ( <column_name>[, <column_name>[, ...] ] ) [ <match_type> ] ]
```

**table_name**

Specifies the table on which the data statistics are defined.

```xml
<table_name> ::= [ [ <database_name>.]<schema_name>.]<identifier>
```

For linked database, `<database_name>` is the name of the remote source. For all other cases, `<database_name>` is the name of the database where the table is located.

**column_name**

Specifies the column for which the data statistics are defined.

```xml
<column_name> ::= <identifier>
```

If no `<column_name>` is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

**match_type**

Controls which data statistics objects to match to `<data_sources>`.

```xml
<match_type> ::= EXACT | CASCADE
```

If `<match_type>` is not specified, then any data statistics object(s) that reference all or some of the columns, but no other columns specified in `<data_sources>` are refreshed.

Specify EXACT to refresh data statistics objects that precisely match `<data_sources>` (including column order).

Specify CASCADE to refresh data statistics objects that reference at least one column in `<data_sources>`.

Use this table to understand how matching is performed based on `<match_type>` when `<data_sources>` is T(A, B, C):

<table>
<thead>
<tr>
<th>Match type</th>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
</table>
| (not specified)     | T(A,C)          | T(A,X) - because T(X) is not a column in `<data_sources>`.
|                     | T(C)            |                     |
|                     | T(B,A)          |                     |
| EXACT               | T(A,B,C)        | T(B,A,C) - because the column order is different than the column order of `<data_sources>`.
|                     |                 | T(A) - because it does not contain the exact same columns and column order of `<data_sources>`.
<p>|</p>
<table>
<thead>
<tr>
<th>Match type</th>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASCADE</td>
<td>T(A,C)</td>
<td>T(X) - because T(X) is not a column in &lt;data_sources&gt;.</td>
</tr>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A,B,C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A,X)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(C,B,A,X)</td>
<td></td>
</tr>
</tbody>
</table>

**match_properties**

Specifies properties to use for matching when selecting data statistics.

```
<match_properties> ::= <match_property>[...]
<match_property> ::= TYPE <data_statistics_type>  | REFRESH TYPE <refresh_type_filter>
```

If TYPE is not specified, then all data statistics objects of any type on the specified data sources are refreshed (ALL). For descriptions of the supported data statistics types see the CREATE STATISTICS Statement topic.

**data_statistics_type**

Specifies the type of data statistics objects to match when selecting the data statistics.

```
<data_statistics_type> ::= TYPE <type_name>
<type_name> ::= HISTOGRAM | SIMPLE | TOPK | SKETCH | SAMPLE | RECORD COUNT | ALL
```

**refresh_type_filter**

Specifies the refresh strategy to match in the data statistics objects when selecting the data statistics to refresh. ALL is the default.

```
<refresh_type_filter> ::= AUTO | MANUAL | ALL
```
Description

The REFRESH STATISTICS statement rebuilds data statistics objects that approximate the specified `<data_sources>`, and only affects ENABLED data statistics objects.

If you employ a manual delta merge strategy instead of an automatic delta merge strategy, consider refreshing your data statistics objects more frequently to avoid stale statistics.

For descriptions of the supported data statistic types (<data_statistics_type>), see the CREATE STATISTICS statement.

Permissions

One of the following is true:

- You own the object you are refreshing statistics on.
- You have the ALTER privilege on the object you are refreshing statistics on.
- For linked database, you have the LINKED DATABASE object level privilege on the remote source, regardless of who owns the remote source.

Example

The following example refreshes the HISTOGRAM on column T(X), where T is a virtual table:

```
REFRESH STATISTICS ON MYSCHEMA.T(X) TYPE HISTOGRAM;
```

The following example refreshes any SKETCH with `<data_sources>` containing one or more of columns T(X), T(Y), or T(Z) (only) in any order:

```
REFRESH STATISTICS ON MYSCHEMA.T(X, Y, Z) TYPE SKETCH;
```

The following example refreshes the database statistics object named HISTOGRAM_T_X:

```
REFRESH STATISTICS MYSCHEMA.HISTOGRAM_T_X;
```

The following example refreshes any data statistic object with `<data_sources>` defined as T(A, B). HISTOGRAMS on T(A) and on T(B) would not be affected.

```
REFRESH STATISTICS ON MYSCHEMA.T(A, B) EXACT;
```

The following example refreshes any data statistic object that has at least one of the table columns. For example, SKETCH on T(A,X) will be refreshed.

```
REFRESH STATISTICS ON MYSCHEMA.T(A, B, C) CASCADE TYPE ALL;
```
The following example refreshes the HISTOGRAM MYSHEMA.MYHISTOGRAM on table column MYSHEMA.T(A):

```
REFRESH STATISTICS MYSHEMA.MYHISTOGRAM ON MYSHEMA.T(A) TYPE HISTOGRAM REFRESH TYPE MANUAL;
```

The following example refreshes the record count on table T:

```
REFRESH STATISTICS ON T TYPE RECORD COUNT;
```

Related Information

CREATE STATISTICS Statement (Data Definition) [page 815]
ALTER STATISTICS Statement (Data Definition) [page 484]
DROP STATISTICS Statement (Data Definition) [page 974]
M_DATA_STATISTICS System View [page 1851]
M_SYSTEM_DATA_STATISTICS System View [page 2189]
DATA_STATISTICS System View [page 1538]

4.10.1.140 REFRESH VIEW Statement (Data Definition)

Refreshes an anonymized view.

Syntax

```
REFRESH VIEW <anonymized_view_name> ANONYMIZATION
```

Syntax Elements

- **anonymized_view_name**
  
  Specifies the name of the anonymized view to refresh.

Description

You must refresh a view before you can select from it.
Refreshing a view can fail if privacy constraints cannot be satisfied due to the source data and parameters of the view. If this occurs, you can alter the view to adjust the constraints.

For more information on data anonymization, see the SAP HANA Administration Guide.

Example

The following example refreshes a fictitious anonymized view called Illness_K_Anon:

```
REFRESH VIEW Illness_K_Anon ANONYMIZATION;
```

For a working example of how to refresh an anonymized view, see the anonymized view example found in the Examples section of the CREATE VIEW statement topic.

Related Information

CREATE VIEW Statement (Data Definition) [page 904]
ALTER VIEW Statement (Data Definition) [page 667]
ANONYMIZATION_VIEWS System View [page 1495]
ANONYMIZATION_VIEW_COLUMNS System View [page 1496]
M_ANONYMIZATION_VIEWS System View [page 1745]
SAP HANA Data Anonymization

4.10.1.141 RELEASE SAVEPOINT Statement (Transaction Management)

Releases a specified savepoint name.

**Syntax**

```
RELEASE SAVEPOINT <savepoint_name>
```

**Syntax Elements**

`savepoint_name` Specifies the case-insensitive identifier that was specified on a SAVEPOINT statement within the current transaction.
**Description**

Releases an established savepoint within the current transaction. The savepoint name specified must exist for the current transaction.

The `<savepoint_name>` identifier can be used in a SAVEPOINT or ROLLBACK TO SAVEPOINT statement.

**Examples**

Release a SAVEPOINT named savepoint1.

```
RELEASE SAVEPONT savepoint1;
```

**Related Information**

ROLLBACK TO SAVEPOINT Statement (Transaction Management) [page 1102]
SAVEPOINT Statement (Transaction Management) [page 1103]

---

**4.10.1.142 RENAME COLUMN Statement (Data Definition)**

Changes the name of a column.

**Syntax**

```
RENAME COLUMN <table_name>.<old_column_name> TO <new_column_name>
```

**Syntax Elements**

- `table_name`
  - Specifies the name of the table where the column is to be renamed.
  - `<table_name> ::= <identifier>`

- `old_column_name`
  - Specifies the old column name.
  - `<old_column_name> ::= <identifier>`
**new_column_name**

Specifies the new column name.

```sql
<new_column_name> ::= <identifier>
```

**Description**

Changes the name of a column.

**Example**

Create table `tab` with two columns named `A` and `B`.

```sql
CREATE ROW TABLE tab (A INT PRIMARY KEY, B INT);
```

Display the column names for table `tab` stored in the SAP HANA database.

```sql
SELECT COLUMN_NAME, POSITION FROM TABLE_COLUMNS WHERE SCHEMA_NAME = CURRENT_SCHEMA AND TABLE_NAME = 'tab' ORDER BY POSITION;
```

Rename column `A` to `C`.

```sql
RENAME COLUMN tab.A TO C;
```

Display the column names for table `tab` after the renaming.

```sql
SELECT COLUMN_NAME, POSITION FROM TABLE_COLUMNS WHERE SCHEMA_NAME = CURRENT_SCHEMA AND TABLE_NAME = 'tab' ORDER BY POSITION;
```

**Related Information**

Introduction to SQL [page 27]
4.10.1.143 RENAME DATABASE Statement (Tenant Database Management)

Changes the name of a tenant database.

**Syntax**

```
RENAME DATABASE <current_database_name> TO <new_database_name>
```

**Syntax Elements**

- **current_database_name**
  Specifies the current database name with an optional schema name.

  `<current_database_name> ::= <identifier>`

- **new_database_name**
  Specifies the new database name.

  `<new_database_name> ::= <identifier>`

**Description**

The RENAME DATABASE statement changes the name of a tenant database.
Renaming tenant databases is not supported on systems running SAP HANA dynamic tiering.

**Permissions**

You must be in SYSTEMDB and have the DATABASE ADMIN privilege to change the name of a tenant database.

**Example**

The following example renames the tenant database MYOPS to MYOPSDEV.

```
RENAME DATABASE MYOPS TO MYOPSDEV;
```
4.10.1.144  RENAME INDEX Statement (Data Definition)

Changes the name of an index.

Syntax

```
RENAME INDEX <current_index_name> TO <new_index_name>
```

Syntax Elements

- **current_index_name**
  
  Specifies the current index name with an optional schema name.
  
  ```
  <current_index_name> ::= [<schema_name>.]<identifier>
  <schema_name> ::= <unicode_name>
  ```

- **new_index_name**
  
  Specifies the new index name.
  
  ```
  <new_index_name> ::= <identifier>
  ```

Description

The RENAME INDEX statement changes the name of an index.

Example

Create Table B and then index idx on column B of table B.

```
CREATE TABLE B (A INT PRIMARY KEY, B INT);
CREATE INDEX idx on B(B);
```

Show the list of index names on table B by using the INDEXES system table.

```
SELECT INDEX_NAME FROM INDEXES WHERE SCHEMA_NAME = CURRENT_SCHEMA AND TABLE_NAME = 'B';
```

Rename index idx to new_idx.

```
RENAME INDEX idx TO new_idx;
```
Show a list of index names on table B after the renaming has occurred.

```
SELECT INDEX_NAME FROM INDEXES WHERE SCHEMA_NAME = CURRENT_SCHEMA AND TABLE_NAME = 'B';
```

Related Information

INDEXES System View [page 1585]
INDEX_COLUMNS System View [page 1587]

4.10.1.145  RENAME SCHEMA Statement (Data Definition)

Rename a schema without creating a new schema.

Syntax

```
RENAME SCHEMA <source_schema_name> TO <target_schema_name>
```

Syntax Elements

*source_schema_name*

Specifies the current name of the schema that you want to rename.

```
<source_schema_name> := <unicode_name>
```

*target_schema_name*

Specifies the new name of the schema.

```
<target_schema_name> := <unicode_name>
```

Attempting to use an existing schema name returns an error.

Description

Rename an existing schema. The schema name of all objects in the source schema is also renamed. All privileges granted to schema objects, data contained in those objects, the validity of the objects, and object IDs are unaffected by the rename.
You cannot rename a user’s default schema.

Permissions

You must be the owner of the source schema and you must have the CREATE SCHEMA system privilege.

Example

Execute the following statements to create two schemas, named schema0 and schema1:

```
CREATE SCHEMA schema0;
CREATE SCHEMA schema1;
```

Create and populate a table in schema1 and create views based on that table:

```
CREATE TABLE schema1.tb1(x, INT);
INSERT INTO schema1.tb1 VALUES (1);
CREATE VIEW schema1.vi1 AS SELECT * FROM schema1.tb1;
CREATE VIEW schema0.vi0 AS SELECT * FROM schema1.vi1;
```

Execute the following statement to rename schema1 to schema2:

```
RENAME SCHEMA schema1 TO schema2;
```

Execute the following SELECT statement:

```
SELECT * FROM schema2.tb1;
```

The statement executes successfully and returns 1 as a result, since the table schema1.tb1 has been renamed to schema2.tb1.

Execute the following SELECT statement:

```
SELECT * FROM schema2.vi1;
```

The statement executes successfully and returns 1 as a result, since the view schema1.vi1 has been renamed to schema2.vi1.

Execute the following SELECT statement:

```
SELECT * FROM schema0.vi0;
```

The statement returns an error indicating that it is invalid as the view schema0.vi0 relies on the view schema1.vi1, which has been renamed to schema2.vi2.

Related Information

CREATE SCHEMA Statement (Data Definition) [page 807]
4.10.1.146  RENAME TABLE Statement (Data Definition)

Changes the schema and/or the name of a table.

Syntax

```
RENAME TABLE <current_table_name> TO <new_table_name>
```

Syntax Elements

- **current_table_name**
  Specifies the current name of the table, with optional schema name.
  
  `<current_table_name> ::= [<schema_name>.]<identifier>`

- **new_table_name**
  Specifies a new table name and/or schema for `<current_table_name>`.
  
  `<new_table_name> ::= [<schema_name>.]<identifier>`

If a table with the same name as `<new_table_name>` already exists in `<schema_name>`, or if `<schema_name>` doesn't exist, an error is returned.

Description

The RENAME TABLE statement changes the schema and/or the name of a table.

**Note**

Renaming column tables with the NO LOGGING option enabled is currently not supported.

Example

Example 1 - Renaming a table in the current schema
Create table A in the current schema.

```
CREATE COLUMN TABLE A (A INT PRIMARY KEY, B INT);
```

Show a list of table names in the current schema.

```
SELECT TABLE_NAME FROM TABLES WHERE SCHEMA_NAME = CURRENT_SCHEMA ORDER BY TABLE_NAME;
```

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Rename table A to a new name B.

```
RENAME TABLE A TO B;
```

Show a list of table names in current schema after the renaming process.

```
SELECT TABLE_NAME FROM TABLES WHERE SCHEMA_NAME = CURRENT_SCHEMA ORDER BY TABLE_NAME;
```

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Example 2 - Renaming a table in another schema

Create schema mySchema and then create a table A in the new schema.

```
CREATE SCHEMA mySchema;
CREATE COLUMN TABLE mySchema.A (A INT PRIMARY KEY, B INT);
```

Show the list of table names in schema mySchema.

```
SELECT TABLE_NAME FROM TABLES WHERE SCHEMA_NAME = 'MYSCHEMA' ORDER BY TABLE_NAME;
```

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
</tbody>
</table>

Rename table A in mySchema to B.

```
RENAME TABLE mySchema.A TO B;
```

Show a list of table names in schema mySchema after the renaming process.

```
SELECT TABLE_NAME FROM TABLES WHERE SCHEMA_NAME = 'MYSCHEMA' ORDER BY TABLE_NAME;
```

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
</tbody>
</table>

Example 3 - Change the schema of the table
Create schema mySchema2.

```
CREATE SCHEMA mySchema2;
```

Change the schema of table B from mySchema to mySchema2.

```
RENAME TABLE mySchema.B TO mySchema2.B;
```

Show a list of table names in the schema mySchema2.

```
SELECT TABLE_NAME FROM TABLES WHERE SCHEMA_NAME = 'MYSCHEMA2' ORDER BY TABLE_NAME;
```

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
</tbody>
</table>

### 4.10.1.147 REPLACE Statement (Data Manipulation)

Updates rows in a table or inserts new rows. This UPSERT and REPLACE statements are synonymous and have identical syntax and purpose.

#### Syntax

```
REPLACE <table_name> [ <column_list_clause> ]
{
  <value_list_clause> [ WHERE <condition> | WITH PRIMARY KEY ]
  | <subquery>
}
```

#### Syntax Elements

**table_name**

Specifies the table to update, with the optional schema name.

```
<table_name> ::= [ [ <database_name>.]<schema_name>.]<identifier>  
<schema_name> ::= <unicode_name>
```

For linked database, `<database_name>` is the name of the remote source. `<identifier>` is the name of the table on the remote source.

**column_list_clause**

Specifies a list of column identifiers, ordered in the order of values in the `<value_list_clause>` or `<subquery>`.

```
<column_list_clause> ::= ( <column_name> [,…] )
```
value_list_clause
Specifies a list of values, or expressions evaluating to values, to be inserted or updated into the table.

<value_list_clause> ::= VALUES ( <expression> [,…] )

WHERE condition
Specifies the conditions where the command should be performed.

<condition> ::=    <condition> OR <condition>   | <condition> AND <condition>   | NOT <condition>   | ( <condition> )   | <predicate>

WITH PRIMARY KEY
Uses the primary key value in the <value_list_clause> to define the row to act upon.

subquery
For information on subqueries, see the SELECT statement.

Description
When this command is used without a subquery, it functions in a similar way to the UPDATE statement. The difference with this command is that when the WHERE clause condition is false it inserts a new record into the table.

When this command is used with a table that has a primary key, the primary key column must be included in the column list. Columns defined with NOT NULL and without a default specification also have to be included in the column list. Columns that are not specified are filled with a default value or NULL.

The UPSERT (or REPLACE) statement with a subquery operates like the INSERT statement. The exception with this command is that, if an existing row in the table has the same primary key value as a new row, the row is updated with the returned record from the subquery. If the table does not have a primary key, then the command functions in an equivalent way to an INSERT statement as an index cannot be used to check for row duplication.

Examples
See the UPSERT statement for examples of how to update rows using the REPLACE/UPSERT statements.

Related Information
SELECT Statement (Data Manipulation) [page 1104]
4.10.1.148 REVOKE Statement (Access Control)

Revoke roles or privileges for the specified objects from a user or role.

**Syntax**

```
REVOKE <system_privilege>,... FROM <grantee>
REVOKE <source_privilege>,... ON REMOTE SOURCE <source_name> FROM <grantee>
REVOKE <schema_privilege>,... ON SCHEMA <schema_name> FROM <grantee>
REVOKE <object_privilege>,... ON <object_name> FROM <grantee>
REVOKE <column_key_privilege> ON CLIENTSIDE ENCRYPTION COLUMN KEY <key_name> FROM <grantee>
REVOKE <role_name>,... FROM <grantee> [ GRANTED BY <grantor> ]
REVOKE STRUCTURED PRIVILEGE <privilege_name> FROM <grantee>
```

**Syntax Elements**

For the definition of all the other syntax elements not listed here, see the GRANT statement.

- **GRANTED BY grantor**
  
  Allows a user with the ROLE ADMIN privilege to revoke a role that they did not grant. `<grantor>` specifies the user who granted `<role_name>` to `<grantee>`.

**Permissions**

Only the user that granted a specific privilege can revoke it.

For users without ROLE ADMIN privileges, only the user that granted a specific role can revoke it.
Description

If a user has been granted a role, then you cannot revoke a subset of the privileges belonging to that role. To grant a subset of privileges contained within a role, revoke the entire role and grant the required privileges separately.

Revoking a privilege or a role can lead to some views becoming inaccessible or procedures that the user can no longer execute. This situation occurs if a view or a procedure depends on a revoked privilege or on one of the privileges the role had.

Revoking a privilege that had been granted with WITH GRANT OPTION or with WITH ADMIN OPTION results in recursive revocation of privileges. Privileges are not only revoked from the user specified in the command, but also from all the users and roles that were granted that privilege directly or indirectly by the specified user.

Privileges can be granted to a user or role by more than one grantor; revocation of a privilege by one grantor user does not necessarily mean that the privilege is revoked from the user or role.

Example

Create a new user named `worker`.

```
CREATE USER worker PASSWORD His_Password_1;
```

Create a new role named `role_for_work_on_my_schema`.

```
CREATE ROLE role_for_work_on_my_schema;
```

Create a new schema named `my_schema`.

```
CREATE SCHEMA my_schema OWNED BY system;
```

Create a new table named `work_done` in the schema.

```
CREATE ROW TABLE my_schema.work_done (t TIMESTAMP, user NVARCHAR (256), work_done VARCHAR (256));
```

Grant the SELECT privilege on any object privilege in `my_schema` to the role `role_for_work_on_my_schema`.

```
GRANT SELECT ON SCHEMA my_schema TO role_for_work_on_my_schema;
```

Grant the INSERT privilege for the `work_done` table to the role `role_for_work_on_my_schema`.

```
GRANT INSERT ON my_schema.work_done TO role_for_work_on_my_schema;
```

Grant the `role_for_work_on_my_schema` role to the `worker` user.

```
GRANT role_for_work_on_my_schema TO worker;
```

Grant TRACE ADMIN privilege for the `worker` user.

```
GRANT TRACE ADMIN TO worker WITH ADMIN OPTION;
```
Grant the DELETE privilege from the `work_done` table to the `worker` user.

```
GRANT DELETE ON my_schema.work_done TO worker WITH GRANT OPTION;
```

Revoke from the role `role_for_work_on_my_schema` the privilege to select from `my_schema`.

```
REVOKE SELECT ON SCHEMA my_schema FROM role_for_work_on_my_schema;
```

Revoke TRACE ADMIN privilege from the `worker` user.

```
REVOKE TRACE ADMIN FROM worker;
```

Revoke the LINKED DATABASE privilege from user `myuser1` using remote source `myremotesys`.

```
REVOKE LINKED DATABASE ON REMOTE SOURCE myremotesys FROM myuser1;
```

Revoke the USAGE privilege on encryption key `Key5` from user `bsmith`.

```
REVOKE USAGE ON CLIENTSIDE ENCRYPTION COLUMN KEY Key5 FROM bsmith;
```

Revoke the CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege from user `bsmith` on schema `my_schema`.

```
REVOKE CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN ON SCHEMA my_schema FROM bsmith;
```

### Related Information

- GRANT Statement (Access Control) [page 1010]
- CREATE USER Statement (Access Control) [page 893]
- ALTER USER Statement (Access Control) [page 654]
- USERS System View [page 1698]
- GRANT Statement (Access Control) [page 1010]
- USERS System View [page 1698]
- ROLES System View [page 1646]
- GRANTED_ROLES System View [page 1581]
- GRANTED_PRIVILEGES System View [page 1580]
- SAP HANA Security Guide
4.10.1.149  ROLLBACK Statement (Transaction Management)

Changes made during the current transaction are reverted and current database session is set to an idle state.

Syntax

ROLLBACK

Description

The SAP HANA database supports transactional consistency which guarantees that a transaction is completely applied to the system or disposed. During a transaction, data manipulation language (DML) modifications to the database can be explicitly reverted via ROLLBACK command. After ROLLBACK is issued, changes made during the current transaction are reverted and current database session is set to an idle state. ROLLBACK only works with an autocommit disabled session.

If you attempt to use the ROLLBACK statements in an autocommit enabled session, then nothing occurs as transactions are automatically committed to the database.

For more information on changing the autocommit status of your database session, see the information about customizing the Administration Console in the SAP HANA Administration guide.

Example

Before attempting to execute the example below, ensure that you are using an autocommit disabled session.

Create a table T.

CREATE ROW TABLE T (KEY INT PRIMARY KEY, VAL INT);

Insert three rows into table T.

INSERT INTO T VALUES (1, 1);
INSERT INTO T VALUES (2, 2);
INSERT INTO T VALUES (3, 3);

Roll back the current transaction.

ROLLBACK;

Select the data in table T.

SELECT * FROM T;
The SELECT command above returns an empty table. The data definition language (DDL) command used to create the table persisted, but the DML used to create the table data has been rolled back.

**Related Information**

SAP HANA Database Backup and Recovery

### 4.10.1.150 ROLLBACK TO SAVEPOINT Statement (Transaction Management)

Rolls back a transaction to the named savepoint without terminating the transaction.

**Syntax**

```sql
ROLLBACK TO SAVEPOINT <savepoint_name>
```

**Syntax Elements**

- `savepoint_name`: Specifies the case-insensitive identifier that was specified on a SAVEPOINT statement within the current transaction.

**Description**

The ROLLBACK TO SAVEPOINT statement undoes any changes that have been made since the SAVEPOINT was established without terminating the current transaction. Changes made before the SAVEPOINT was set are not undone. Savepoints set after that savepoint to which you rolled back are erased, but the rollback savepoint remains.

The savepoint name specified must exist within the current transaction.

A ROLLBACK or COMMIT erases all savepoints.

The `<savepoint_name>` identifier can be used in a SAVEPOINT or RELEASE SAVEPOINT statement.
Examples

Create a table T.

```
CREATE ROW TABLE T (KEY INT PRIMARY KEY, VAL INT);
```

Insert two rows into table T.

```
INSERT INTO T VALUES (1, 1);
INSERT INTO T VALUES (2, 2);
```

Set a savepoint.

```
SAVEPOINT savepoint1;
```

Insert a row into table T.

```
INSERT INTO T VALUES (3, 3);
```

Roll back the current transaction to savepoint1.

```
ROLLBACK TO SAVEPOINT savepoint1;
```

Select the data in table T.

```
SELECT * FROM T;
```

The SELECT command above rows one and two, but not three.

Related Information

- RELEASE SAVEPOINT Statement (Transaction Management) [page 1087]
- SAVEPOINT Statement (Transaction Management) [page 1103]

4.10.1.151  SAVEPOINT Statement (Transaction Management)

Defines a location to which a transaction can return if part of the transaction is conditionally canceled.

Syntax

```
SAVEPOINT <savepoint_name>
```
Syntax Elements

savepoint_name Specifies a unique incase-sensitive identifier for the savepoint.

Description

Establish a savepoint within the current transaction.

The number of active savepoints for each transaction is unlimited. Savepoint names must be unique within a given transaction. If you create a second savepoint with the same identifier as an earlier savepoint, then the earlier savepoint is erased.

The <savepoint_name> identifier can be used in a RELEASE SAVEPOINT or ROLLBACK TO SAVEPOINT statement.

This statement should not be confused with the ALTER SYSTEM SAVEPOINT statement, which performs a database checkpoint, not a transactional checkpoint.

Examples

Create a SAVEPOINT named savepoint1.

```sql
SAVEPOINT savepoint1;
```

Related Information

RELEASE SAVEPOINT Statement (Transaction Management) [page 1087]
ROLLBACK TO SAVEPOINT Statement (Transaction Management) [page 1102]
ALTER SYSTEM SAVEPOINT Statement (System Management) [page 567]

4.10.1.152 SELECT Statement (Data Manipulation)

Queries data from the database.

Syntax

```sql
<select_statement> ::=      { [ <with_clause> [page 1105] ] <subquery> [ <select_option_1> ] [ <collation_clause> ] [ <hint_clause> [page 1110] ]
```
### Syntax Elements

**with_clause**

Specifies an output of the subquery to be stored in a temporary result set named `<query_name>`.

```
<with_clause> ::= WITH <with_list_element> [ {, <with_list_element> }... ]
<with_list_element> ::= <query_name> [ <column_list_clause> ] AS ( <subquery> )
<query_name> ::= <identifier>
<column_list_clause> ::= ( <column_name>, ... )
<column_name> ::= <identifier>
```

The query name is an alias of the `<subquery>` and it should be unique within the `<with_clause>`. The `<with_clause>` cannot support recursive query expressions.

The list of column identifiers is ordered according to the order of values in the `<subquery>`. The column list can be omitted.

**select_option_1**

```
<select_option_1> ::= 
  { <for_update_clause> [page 1105] 
    <for_share_lock_clause> [page 1106] 
    <time_travel_clause> [page 1107] 
    <for_system_time_clause> } 
```

**for_update_clause**

Locks the selected records so that other users cannot lock or change the records until the end of this transaction.

```
<for_update_clause> ::= 
```
FOR UPDATE supports row, column, and virtual tables created from HANA remote sources and can be used with either one or more column or row tables. If the `<update_column_name_list>` list does not contain a primary key column, the non-key exclusive lock mode will be used.

**Note**

In case you use multiple row tables, they all need to have one and the same location.

If a HANA query using virtual tables cannot be entirely delegated to the remote source, the query returns the error message indicating that the FOR UPDATE feature is not supported.

A table used in a subquery cannot be locked. Also only one table/view can be currently locked. The lock is released when the corresponding transaction is finished by commit or rollback.

**update_column_name_list**

Specifies the columns within the specified table or view in the FROM clause to be locked.

```
<update_column_name_list> ::= ( <column_name> [, <column_name> ] [ ,... ] )
```

**wait_nowait**

Specifies when to return an error if a lock cannot be obtained on a record.

```
<wait_nowait> ::=      { WAIT <unsigned_integer> | NOWAIT }
```

<wait_nowait> is not supported when querying virtual tables.

If WAIT `<unsigned_integer>` is specified, then the statement waits up to the specified number of seconds for each record. If the statement fails to acquire a lock after waiting, then it returns an error message indicating a timeout. If NOWAIT is specified, and the statement fails to acquire record locks, then an error is returned indicating that the resource is busy. WAIT 0 is equivalent to NOWAIT. If `<wait_nowait>` is not specified, then the default behavior reflects the lock_wait_timeout transaction timeout setting located in indexserver.ini.

**IGNORE LOCKED**

Specifies to ignore locked records.

**for_share_lock_clause**

Specifies the columns which should be protected against concurrent updates.

```
<for_share_lock_clause> ::=      FOR SHARE LOCK [ OF <update_column_name_list> ]     [ <wait_nowait> ] [ IGNORE LOCKED ]
```

FOR SHARE LOCK is not supported when querying virtual tables.

When OF `<column_name_list>` is used, it is only guaranteed that the specified columns in the list stay unmodified (by other transactions). It isn’t guaranteed that the record stays intact/unmodified.
Acquires shared locks on the queried records. By doing so, the locked records stay intact until the transaction is committed or rolled back. If a queried record is already exclusively locked by another transaction, the SELECT FOR SHARE LOCK statement waits for the lock to be released, or waits for the current transaction’s lock-wait-timeout value, whichever is shorter.

**update_column_name_list**

Specifies the columns within the specified table or view in the FROM clause to be locked.

```sql
<update_column_name_list>> ::= ( <column_name> [, <column_name> [,...] ] )
```

**wait_nowait**

Specifies when to return an error if a lock cannot be obtained on a record.

```sql
<wait_nowait> ::=      { WAIT <unsigned_integer> | NOWAIT }
```

<wait_nowait> is not supported when querying virtual tables.

If WAIT <unsigned_integer> is specified, then the statement waits up to the specified number of seconds for each record. If the statement fails to acquire a lock after waiting, then it returns an error message indicating a timeout. If NOWAIT is specified, and the statement fails to acquire record locks, then an error is returned indicating that the resource is busy. WAIT 0 is equivalent to NOWAIT. If <wait_nowait> is not specified, then the default behavior reflects the <lock_wait_timeout> transaction timeout setting located in indexserver.ini.

**IGNORE LOCKED**

Specifies to ignore records already locked. IGNORE LOCKED is not supported when querying virtual tables.

**time_travel_clause**

Specifies a time frame for the query.

```sql
<time_travel> ::=      { <commit_id> | UTCTIMESTAMP <timestamp> }
```

Keywords related to time travel can be used for statement-level time travel to go back to the snapshot specified by <commit_id> or UTCTIMESTAMP.

Time travel is possible only with a history column table. <commit_id> can be obtained from the M_HISTORY_INDEX_LAST_COMMIT_ID system view after each commit, and its related <timestamp> can be obtained from the TRANSACTION_HISTORY system view.

**for_system_time_clause**

Selects records that were active within the specified validity period. This option is only for use with querying system-versioned tables; if the target is a non-history table, the statement fails. If the target is a view which doesn’t access any history table, the <for_system_time_clause> specification is ignored.

```sql
<for_system_time_clause> ::=      { as_of_spec}
```
as_of_spec Returns records that were active as of the specified timestamp.

```sql
<as_of_spec> ::= FOR SYSTEM_TIME AS OF '<timestamp1>'
```

from_to_spec Returns records that were active any time within the specified time range, even if their start time was before `<timestamp1>` or their end time was after `<timestamp2>`.

```sql
<from_to_spec> ::= FOR SYSTEM_TIME FROM '<timestamp1>' TO '<timestamp2>'
```

Records that stopped being active right at `<timestamp1>` are not included, nor are records that started being active right at `<timestamp2>`.

between_spec Returns records that were active any time within the specified time range, even if their start time was before `<timestamp1>` or their end time was after `<timestamp2>`.

```sql
<between_spec> ::= FOR SYSTEM_TIME BETWEEN '<timestamp1>' AND '<timestamp2>'
```

But unlike `<from_to_spec>`, it also returns records that started being active right at `<timestamp2>`.

for_json_clause Returns a results set as a JSON document.

```sql
<for_json_clause> ::= FOR JSON [ ( <json_option_string_list> ) ] [ <json_return_clause> ]
```

The `<for_json_clause>` clause is supported in table subqueries (for example, `SELECT...FROM (SELECT...FROM myTable FOR JSON)`) and scalar subqueries (for example, `SELECT (SELECT... FROM myTable FOR JSON) FROM s`), as well inside `INSERT INTO` and `FOR UPDATE` subqueries.

json_option_string_list Sets options for the resulting JSON

```sql
<json_option_string_list> ::= '<json_basic_string_expression>' [,...]
```

A `json_basic_string_expression` is a comma-separated string of option-value pairs. The following table defines the options and what they control.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>format</td>
<td>Whether to insert newline and tab for elements. Possible values are 'yes' or 'no' (the default).</td>
</tr>
<tr>
<td>omitnull</td>
<td>Whether to omit NULL value records in the resulting JSON document. Possible values are 'yes' (the default), or 'no'.</td>
</tr>
</tbody>
</table>
Optionwrap

Whether to include [ ] at the beginning and end of the JSON document. Possible values are 'yes' (the default), or 'no'.

json_return_clause

Specifies the return type of the results. The default is NCLOB.

<json_returns_clause> ::=  RETURNS { VARCHAR ( <integer> ) | NVARCHAR ( <integer> ) | CLOB | NCLOB }

for_xml_clause

Returns a result set as an XML document.

<for_xml_clause> ::= FOR XML [ ( <xml_option_string_list> ) ] [ <xml_returns_clause> ]

The <for_xml_clause> clause is supported in table subqueries (for example, SELECT...FROM (SELECT...FROM mytable FOR XML);) and scalar subqueries (for example, SELECT (SELECT... FROM myTable FOR XML) FROM s;), as well inside INSERT INTO and FOR UPDATE subqueries. The query may return invalid XML if column names do not obey XML naming rules.

xml_option_string_list

Sets options for the resulting XML.

<xml_option_string_list> ::= '<xml_basic_string_expression>' [,...]

<xml_basic_string_expression> is a comma-separated string of option-value pairs (for example, AS OF COMMIT ID FOR XML (‘schemaloc’=’http:/ /thiscompany.com/schemalib’, ‘targetns’=’http:/ /thiscompany.com/samples’, ‘statement’=’yes’)). The following table defines the options and what they control.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnstyle</td>
<td>The representation of SQL columns. Possible values are 'element' (the default), or 'attribute'.</td>
</tr>
<tr>
<td>format</td>
<td>Whether to insert newline and tab for elements. Possible values are 'yes' (the default), or 'no'.</td>
</tr>
<tr>
<td>header</td>
<td>Whether to include the XML declaration header (&lt;xml version=&quot;1.0&quot;?&gt;). Possible values are 'yes', or 'no' (the default).</td>
</tr>
<tr>
<td>incremental</td>
<td>Whether to return a single row or multiple rows. Possible values are 'yes', or 'no' (the default).</td>
</tr>
</tbody>
</table>
### Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nullstyle</td>
<td>Whether to omit or present a null attribute for NULL values. Possible values are 'omit' (the default), or 'attribute'.</td>
</tr>
<tr>
<td>root</td>
<td>Whether to include a root element for the table name. Possible values are 'yes' (the default), or 'no'.</td>
</tr>
<tr>
<td>rowname</td>
<td>Sets the name of the row element. The default is 'row'.</td>
</tr>
<tr>
<td>schemaloc</td>
<td>Sets the <code>&lt;schemalocation&gt;</code> value in the root element. If the <code>&lt;root&gt;</code> option is set to no, then this option is ignored. This value is a string with a URI.</td>
</tr>
<tr>
<td>tablename</td>
<td>Names the root element. If the root option is set to 'no', then this option is ignored. The value is a string and the default is 'resultset'.</td>
</tr>
<tr>
<td>targetns</td>
<td>Sets the <code>&lt;targetnamespace&gt;</code> value in the root element. If the <code>&lt;root&gt;</code> option is set to 'no', then this option is ignored.</td>
</tr>
</tbody>
</table>

### xml_returns_clause

Specifies the return type of the results. The default is CLOB.

```
<xml_returns_clause> ::=  RETURNS { VARCHAR ( <integer> ) | NVARCHAR ( <integer> ) | CLOB | NCLOB }
```

### collation_clause

Specifies the collation to use at the statement level. For each language, there are three entries, for example, ENGLISH, ENGLISH_CI, and ENGLISH_AI. The later two are case-insensitive and accent-insensitive.

```
<collation_clause> ::= WITH COLLATION <collation_name>
```

For example, in the following SELECT statement, collation ENGLISH applies to both column a and b for order comparisons.

```
SELECT * FROM T3 ORDER BY a, b WITH COLLATION ENGLISH;
```

Not all collation languages are supported. For a supported list, query the COLLABTIONS system view where IS_BASE_COLLATION = 'TRUE'.

```
SELECT * FROM COLLABTIONS WHERE IS_BASE_COLLATION = 'TRUE';
```

### hint_clause

Specifies a hint to use for the query.

```
<hint_clause> ::=
```
When there are hint clauses in the subquery, only the most outer hint is applied.

Incorrect syntax or semantics generate an SQL error.

**select_clause**

Specifies an output to be returned either to the caller or to an outer `<select_clause>` if one exists.

```sql
<select_clause> ::= SELECT [ TOP <unsigned_integer> ] [ { ALL | DISTINCT } ] <select_list>
```

**TOP**

Specifies that the first `<unsigned_integer>` records from the SQL statement should be returned.

To ensure consistent, predictable results from the TOP clause, use the ORDER BY clause to sort the records in a specific order. Without an ORDER BY clause, the order of records is not guaranteed and the results of the TOP clause may change.

**DISTINCT**

Specifies that only one copy of each set of duplicate records selected should be returned.

**ALL**

Specifies that all rows selected, including all copies of duplicates should be returned. This is the default option.

**select_list**

Specifies the select items that are in scope for the query.

```sql
<select_list> ::= <select_item>[ (, <select_item> )... ]
<select_item> ::= <column_name> | <expression> | <association_expression> } [ AS <alias> ] [ <table_name>.] * }
```

**select_item**

Specifies the details of the columns to be retrieved.

**expression**

Combination of one or more values, operators, and SQL functions that evaluate to a value.

**association_expression**

Specifies a column association, with optional filter (condition) on column values.
**association_cardinality** ::= USING { ONE | MANY } TO { ONE | MANY } JOIN

**condition** Applies the specified predicate, filtering the column values. For example myTable[myColumn>100] filters matches values of myColumn that are greater than 100. The square brackets around <condition> are required syntax.

**association_cardinality** Specifies the join to apply.

**parameterized_view_value_list** Specifies the parameter’s value as input when the associated table or associated reference is a parameterized view.

<parameterized_view_value_list> ::= <parameter_view_value> [...]
<parameter_view_value> ::= { <parameter_name> => <value> | <value> }

**alias**

Specifies an alias for the expression.

**table_name**

Specifies the table that is the source for the selected columns, with the optional schema name.

For linked database, <database_name> is the name of the remote source. <identifier> is the name of the table on the remote source.

(*) selects all columns from all tables or views listed in the <from_clause>. If you provide a schema name or a table name with an asterisk (*), then it limits the scope of the result set to the specified table.

**from_clause**

Specifies inputs such as tables, views, and subqueries to be used as the data source for the SELECT statement.

<from_clause> ::= FROM <table_expression> [, <table_expression> ...]
<table_expression> ::= { <table_ref>
  <system_versioned_table_ref>
  <subquery>
  <joined_table>
  <collection_derived_table>
  <function_reference>
  <JSON_collection_table>
  :<table_variable>
  <associated_table_expression> }

**table_expression**

Specifies the data source for the query.

**table_ref**

Specifies a table reference as a data source.

<table_ref> ::= <table_name>
[ [ for_system_time_clause ] | [ for_application_time_period ] ]
[ [ for_partition_restriction ] ]
[ [ AS ] <table_alias> ]
[ <table_sample_clause> ]

**table_name**
Specifies a table as a data source.

```
<table_name> ::= [ [ <database_name>.]<schema.name>.]<identifier>
```

For linked database, `<database_name>` is the name of the remote source. `<identifier>` is the name of the table on the remote source.

**for_application_time_period**

Selects records that were active as of the specified timestamp. This option is only for use with querying application-period time tables.

```
<for_application_time_period> ::= FOR APPLICATION_TIME AS OF '<timestamp>'
```

**partition_restriction**

Restricts the query to the specified partition(s) of a partitioned table.

```
<partition_restriction> ::= PARTITION ( <partition_number> [ {, <partition_number> }...] )
<partition_number> ::= <unsigned_integer>
```

If no partition is specified, then all partitions are the source of the SELECT. `<partition_restriction>` can be obtained from the M_CS_PARTITIONS system view. This syntax is available for partitioned tables only.

**table_alias**

Specifies an alias for the table or table join.

```
<table_alias> ::= <identifier>
```

**tablesample_clause**

Applies the SELECT statement to a random sample of the table data.

```
<tablesample_clause> ::= TABLESAMPLE [ BERNOULLI | SYSTEM ] ( <sample_size> )
<sample_size> ::= <exact_numeric_literal>
```

`<sample_size>` specifies the percentage of the table to be returned as a sample. Values must be greater than 0 and less than or equal to 100. (100 returns the complete table). The goal of the TABLESAMPLE operator is to allow queries to be executed over ad-hoc random samples of tables. Samples are computed uniformly over rows in a columnar base table. Samples can either be uniform random samples (SYSTEM sampling) or uniform and independent random samples (BERNOULLI sampling). SYSTEM sampling exploits the lack of the independence requirement by sampling blocks of rows at a time, while SYSTEM sampling offers performance advantages over BERNOUlli sampling, it comes at the cost of larger variance in sample size.

**subquery**

Specifies a SELECT statement as the data source.

```
<subquery> ::= ( <select_statement> ) [ AS <subquery_alias> ]
<subquery_alias> ::= <identifier>
```

`<subquery_alias>` specifies an alias for the subquery.

**joined_table**
Specifies a table join.

```
<joined_table> ::=  
<table_expression> [ <join_type> ] [ <join_cardinality> ] JOIN 
<table_expression> ON <predicate>  
| <table> CROSS JOIN <table> | <joined_table> 
| <case_join>
```

table

Specifies the table to be used in the join.

join_type

Specifies the join cardinality.

```
<join_type> ::= ( INNER | { LEFT | RIGHT | FULL } [ OUTER ] )
```

- An INNER JOIN returns records that have matching values in both tables.
- A LEFT (OUTER) JOIN returns all records from the left table, and the matched records from the right table.
- A RIGHT (OUTER) JOIN returns all records from the right table, and the matched records from the left table.
- A FULL (OUTER) JOIN returns all records when there is a match in either left or right table.
- OUTER is an optional keyword that applies to LEFT, RIGHT, and FULL; it does not change the semantics of the join syntax.

join_cardinality

Specifies the join cardinality.

Join cardinality specifies how many matching entries in one table exist in the other table in a join operation. This information can be exploited for better query performance.

Ensure that you chose the correct join cardinality by validating that the query returns the expected results; imprecise join cardinality can lead to unexpected results.

```
<join_cardinality> ::= ( MANY TO MANY  
    | MANY TO ONE  
    | MANY TO EXACT ONE  
    | ONE TO MANY  
    | EXACT ONE TO MANY  
    | ONE TO ONE  
    | EXACT ONE TO ONE  
    | ONE TO EXACT ONE  
    | EXACT ONE TO EXACT ONE )
```

ONE means 0 or 1 matching record and EXACT ONE means only 1 matching record. MANY means no restriction and it can be 0 record or more than 1 record. If no information is specified, MANY TO MANY is default which means no restriction.

ON predicate

ON `<predicate>` specifies a join predicate.

CROSS JOIN

CROSS JOIN is another `<join_type>`, but it only allows a `<table>` on the left and right side, so it is on its own line in the syntax. CROSS JOIN produces the cross-product of two tables. In many SQL dialects, including HANA SQL, a comma is semantically equivalent to the CROSSSELECT * from table1, table2...CROSS JOIN specifies that a cross join should be performed. However,
specifying a comma is not recommended because it has a lower precedence than CROSS JOIN and can return unexpected results, especially when the query also involves outer join operators.

**case_join**

Specifies a case join, which allows you to choose a different table to be joined for each record.

\[
\text{case}_\text{join} \] \( ::= \) \( <\text{table}_0> \) LEFT OUTER MANY TO ONE CASE JOIN \( \text{WHEN} \) \( <\text{condition}_\text{predicate}_1> \) \( \text{THEN} \) \( \text{RETURN} \) \( <\text{output}_\text{column}_\text{list}_1> \) FROM \( <\text{table}_1> \) \( \text{ON} \) \( <\text{join}_\text{predicate}_1> \) \( \ldots \) \( \text{WHEN} \) \( <\text{condition}_\text{predicate}_N> \) \( \text{THEN} \) \( \text{RETURN} \) \( <\text{output}_\text{column}_\text{list}_N> \) FROM \( <\text{table}_N> \) \( \text{ON} \) \( <\text{join}_\text{predicate}_N> \) \[ \text{ELSE} \text{RETURN} \) \( <\text{output}_\text{column}_\text{list}_{N+1}> \) FROM \( <\text{table}_{N+1}> \) \( \text{ON} \) \( <\text{join}_\text{predicate}_{N+1}> \) \] \( \text{END} \)

For each row in \( <\text{table}_0> \), the first satisfied \( \text{WHEN} \) clause will be joined with the table in the \( \text{THEN} \) clause.

For the \( \text{WHEN} \) clause, only columns from left side of a case join can be used in the \( \text{THEN} \) clause.

In the \( \text{THEN} \) clause, \( <\text{table}_1> \) represents the joined table name, and \( <\text{output}_\text{column}_\text{list}_1> \) represents the projection columns as a result of the join. \( <\text{table}_1> \) and \( <\text{output}_\text{column}_\text{list}_1> \) must have the same number of columns or an error is returned.

If the \( \text{WHEN} \) clauses do not produce matches and no \( \text{ELSE} \) clause is specified, the join produces NULL columns.

**collection_derived_table**

Specifies a collection derived table as the data source.

\[
\text{collection}_\text{derived}_\text{table} \] \( ::= \) \( \text{UNNEST} \) \( \{ <\text{collection}_\text{value}_\text{expression}> [ <\text{collection}_\text{value}_\text{expression}> ] \ldots \} \) \[ \text{WITH ORDINALITY} \] AS \( <\text{table}_\text{alias}> \) \( <\text{derived}_\text{column}_\text{list}> \)

\[
\text{collection}_\text{value}_\text{expression} \] \( ::= \) \( \text{ARRAY} \) \( \{ <\text{table}_\text{expression}> [ <\text{table}_\text{expression}> ] \ldots \} \) \| \( <\text{column}_\text{name}> \)

\[
\text{derived}_\text{column}_\text{list} \] \( ::= \) \( <\text{column}_\text{name}>, \ldots \)

Collection-derived tables can contain a reference to another element of the \( \text{from}_\text{clause} \) that has been specified before them. This reference has inner join characteristics because the referenced element of the \( \text{from}_\text{clause} \) is used as a parameter to execute the subquery.

Collection values (Array) can be interpreted as the columns of a table by “turning them 90 degrees clockwise”.

You can use an array constructor or a field name of ARRAY type.

\( <\text{derived}_\text{column}_\text{list}> \) specifies a list of column identifiers, ordered in the order of values in the collection value expressions. The column list can be omitted.

**function_reference**

Specifies the results of a table function, including SQLScript user-defined table functions, as the data source.

\[
\text{function}_\text{reference} \] \( ::= \) \( <\text{function}_\text{name}>\{ <\text{proc}_\text{arg}_\text{list}> \) \| \( <\text{opt}_\text{parameter}_\text{key}_\text{value}_\text{list}> \) \}

\[
<\text{function}_\text{name}> \] \( ::= \) \[ [[<\text{database}_\text{name}>.]<\text{schema}_\text{name}>.]<\text{function}_\text{identifier}> \]
<proc_arg_list> and <opt_parameter_key_value_list> are the allowable arguments and parameters for the function.

<database_name> and <schema.name> are for use with SQLScript user-defined table functions. For more information on SQLScript user-defined table functions, see the SAP HANA SQLScript Reference Guide.

JSON_collection_table

Specifies a JSON collection table as the data source. See the SELECT Statement (JSON Document Store) topic for more information on selecting from a JSON collection table.

table_variable

Specifies a SQLScript table variable. See the topic on declarative SQLScript logic in SAP HANA SQLScript Reference Guide for more information about table variables.

associated_table_expression

Specifies a table that is associated with other tables as the data source.

```plaintext
<associated_table_expression> ::= <associated_table_name>
          [ [ <condition> ] ]::<associated_ref>
          [ [ <condition> ] ]
          [ <association_cardinality> ]

<association_cardinality> ::= USING [ ONE | MANY ] TO { ONE | MANY } JOIN
```

<condition> applies the specified predicate to filter the column values. For example, `myTable[myColumn>100]` filters matches values of myColumn that are greater than 100. The square brackets around <condition> are required syntax.

<association_cardinality> specifies the join to apply.

INTO variable_name_list

Selects values into an array of one or more variables.

```plaintext
<variable_name_list> ::= <variable_name> [,<variable_name> [,... ] ]
<variable_name> ::= { <identifier> | <identifier> [ <index> ] | DEFAULT <const_value> }
```

⚠️ Caution

The bold font used for the square brackets around <index> indicate that you must INCLUDE the square brackets when specifying the index (for example, A_COPY[1], B_COPY[1]). In this instance, the brackets do not mean that <index> is optional.

where_clause

Specifies predicates on inputs in the FROM clause.

```plaintext
<where_clause> ::= WHERE <condition>
<condition> ::= <condition> OR <condition>
       | <condition> AND <condition>
       | NOT <condition>
       | ( <condition> )
       <predicate>
```

<predicate> ::=
group_by_clause

Groups the selected rows based on the values in the specified columns.

GROUPING SETS

Generates results with specified multiple groupings of data in a single statement. If no additional options are set, such as BEST and LIMIT, then the result produced is the same as a UNION ALL of the aggregation of each specified group.

For example:

```
SELECT col1, col2, col3, * count(*)
FROM t
GROUP BY GROUPING SETS ( (col1, col2), (col1, col3) );
```
Is equivalent to:

```sql
SELECT * col1, col2, NULL, count(*)
FROM t GROUP BY col1, col2
UNION ALL
SELECT col1, NULL, * col3, count(*)
FROM t GROUP BY col1, col3;
```

In the grouping-sets query, each of (col1, col2) and (col1, col3) specifies the grouping.

If columns are between GROUP BY and grouping sets, then they are common columns of all grouping sets. For example:

```sql
SELECT col1, col2, col3, * count(*)
FROM t
GROUP BY col4, col5, GROUPING SETS ( (col1, col2), (col1, col3) );
```

Is equivalent to:

```sql
SELECT * col1, col2, NULL, count(*)
FROM t GROUP BY col4, col5, col1, col2
UNION ALL
SELECT col1, NULL, * col3, count(*)
FROM t GROUP BY col4, col5, col1, col3;
```

ROLLUP
Generates results with multiple levels of aggregation in a single statement.

For example:

```sql
ROLLUP (col1, col2, col3);
```

Is equivalent to the following statement, with an additional aggregation without grouping.

```sql
GROUPING SETS ( (col1, col2, col3), (col1, col2), (col1) )
```

Thus, the number of grouping sets that result set contains is the number of columns in ROLLUP list plus one for a last aggregation if there is no additional option.

If columns are between GROUP BY and ROLLUP, then the columns are common columns of all grouping sets. For example:

```sql
col4, col5, ROLLUP (col1, col2, col3)
```

Is equivalent to:

```sql
GROUPING SETS ( (col4, col5, col1, col2, col3), (col4, col5, col1, col2),
(col4, col5, col1), (col4, col5) )
```

CUBE
Generates results with multiple levels of aggregations in a single statement.

For example:

```sql
CUBE (col1, col2, col3)
```
Is equivalent to the following statement, with an additional aggregation without grouping.

```sql
GROUPING SETS ( (col1, col2, col3), (col1, col2), (col1, col3), (col2, col3), (col1), (col2), (col3) )
```

Thus, the number of grouping sets that result set contains is the same as all possible permutations of columns in the CUBE list plus one for the last aggregation if there is no additional option.

If columns are between GROUP BY and CUBE, then the columns are common columns of all grouping sets. For example:

```sql
col1, col5, CUBE (col1, col2, col3)
```

is equivalent to:

```sql
GROUPING SETS( (col1, col5, col1, col2, col3), (col1, col5, col1, col2), (col1, col5, col1, col3), (col1, col5, col2, col3), (col1, col5, col1), (col1, col5, col2), (col1, col5, col3), (col1, col5) )
```

Any combination of expression, grouping sets, ROLLUP, and CUBE are possible according to SQL standard. Each of expression, ROLLUP, and CUBE can be converted to a grouping set. Several grouping sets also can be converted to a single grouping set by repeating to combine two successive grouping sets to a single grouping set. Two successive grouping sets are combined to a single grouping set by cross-producing `<grouping_expression_list>`. For example:

```sql
SELECT col1, col2, col3, * count(*)
FROM t
GROUP BY GROUPING SETS( (col1, col2), (col3) ),
        GROUPING SETS( (col4), (col5, col6) );
```

is equivalent to:

```sql
SELECT col1, col2, col3, * count(*)
FROM t
GROUP BY GROUPING SETS( (col1, col2, col4), (col3, col4),
                        (col1, col2, col5, col6), (col3, col5, col6) );
```

**BEST signed_integer**

Returns only the top-n grouping sets sorted in descending order of the number of rows aggregated in each grouping set.

`<signed_integer>` can be zero, positive, and negative. When n is zero, it is the same as the BEST option not being set. When n is negative, it means sorting in ascending order.

Can be specified for a single column table or a single OLAP view.

**LIMIT**

Returns the first `<unsigned_integer>` grouped records after skipping OFFSET `<unsigned_integer>` for each grouping set.

```sql
LIMIT <unsigned_integer> [ OFFSET <unsigned_integer> ]
```

To ensure consistent, predictable results from the LIMIT clause, use the ORDER BY clause to sort the records in a specific order. Without an ORDER BY clause, the order of records is not guaranteed and the results of the LIMIT clause may change.

**WITH SUBTOTAL**
Returns an additional subtotal for each grouping set in the returned results controlled by OFFSET or LIMIT. Unless OFFSET and LIMIT are specified, the value is the same as WITH TOTAL.

WITH BALANCE

Returns for each grouping set an additional aggregated value of the remaining values not returned by OFFSET or LIMIT.

WITH TOTAL

Returns for each grouping set an additional row that is the aggregated total value. OFFSET and LIMIT options do not change this value.

TEXT_FILTER

Performs text filtering or highlighting on the grouping columns using a single-quoted string.

```
  TEXT_FILTER <filterspec>
```

Write logical operators in uppercase characters.

**filterspec**

A filter defined by `<filterspec>` is a token/phrase or tokens/phrases connected with logical operators such as AND, OR, and NOT.

```
<filterspec> ::=      '\[ <prefix> \] <element> <subsequent> [ {, <subsequent> }... ]'
<subsequent> ::= \[<prefix_subsequent>\]<element>
<prefix_subsequent> ::=      { + | - | NOT | AND | AND NOT | OR }
```

A filter is applied only to the first grouping column in each grouping set.

**prefix**

Prefixes + and - mean inclusion (AND) and exclusion (AND NOT), respectively.

```
<prefix> ::= { + | - | NOT }
```

For example, ab -cd is the same as ab AND NOT cd. If FILL UP is not specified, only grouped records that have matching values are returned.

**element**

```
<element> ::=      { <token> | <phrase> }
```

**token** A token matches a string that contains its corresponding word in a case-insensitive manner using unicode letters or digits. For example, ab matches ab cd and cd Ab but does not match abcd. A token can contain the wild card character * that matches any string and ? that matches any character. Inside a phrase, * and ? do not function as wild card characters. With tokens and phrases logical operators AND, OR, and NOT can be used together. As OR is the default operator, ab cd is the same as ab OR cd.

**phrase** A double-quoted string that does not contain double quotations inside.

**FILL UP**
Returns not only matched grouped records, but also non-matched records. TEXT_FILTER is useful to identify which records have been matched. See ‘Related Functions’ below.

SORT MATCHES TO TOP Returns matching values before non-matching values for each grouping set. This option cannot be used with SUBTOTAL, BALANCE and TOTAL.

STRUCTURED RESULT

Returns results as temporary tables. For each grouping set a single temporary table is created. If the WITH OVERVIEW option is set, an additional temporary table is created for the overview of the grouping sets. The names of the temporary tables are specified by the PREFIX option.

WITH OVERVIEW Returns an overview in a separate additional table.

PREFIX prefix_table_name

Specifies a prefix for naming the temporary tables.

<prefix_table_name> is an <identifier> and must start with “#”, which means a temporary table. If PREFIX is not specified, the default prefix is “#GN” followed by a non-negative integer number. See ‘Return Format’ below.

MULTIPLE RESULTSETS

Specifies that results should be returned in multiple result sets.

Related Functions:

- GROUPING_ID(column_name_list) function returns an integer number to identify which grouping set each grouped record belongs to.
- GROUPING(column) function returns 1 if the column is used for grouping. Otherwise it returns 0.
- TEXT_FILTER (<grouping_column>) function, which is used with TEXT_FILTER, FILL UP, and SORT MATCHES TO TOP, displays matching values or NULL. NULL is displayed for non-matching values when FILL UP option is specified.

Return Format:

- If neither STRUCTURED RESULT nor MULTIPLE RESULTSETS is set, then the union result of all grouping sets is returned with NULL values filling up attributes that are not included in a specific grouping set.
- With STRUCTURED RESULT, temporary tables are created additionally, and can be queried using “SELECT * FROM <table_name>” in the same session. The name of the tables follows the form:
  - <PREFIX>0: this table contains the overview if WITH OVERVIEW is specified
  - <PREFIX>n: n-th grouping set subject to re-ordering by the BEST parameter
- With MULTIPLE RESULTSETS, multiple result sets are returned. Grouped records for each grouping set are in a single result set.

having_clause

Selects the specified groups that satisfy the predicates.

<having_clause> ::= HAVING <condition>

If this clause is omitted, all groups are selected.

set_operator

Enables multiple SELECT statements to be combined but return only one result set.

<set_operator> ::=
UNION

[ ALL | DISTINCT ]

INTERSECT [ DISTINCT ]

EXCEPT [ DISTINCT ]

UNION ALL

Selects all records from all SELECT statements. Duplicates are not removed.

UNION [DISTINCT]

Selects all unique records from all SELECT statements by removing duplicates found from different
SELECT statements. UNION has the same function as UNION DISTINCT.

INTERSECT [DISTINCT]

Returns records that exist in all SELECT statement results.

EXCEPT [DISTINCT]

Returns all unique records from the first SELECT statements after removing the duplicates in the
following SELECT statements. MINUS is accepted as a synonym for EXCEPT.

order_by_clause

Sorts records by expressions (such as columns) or by positions.

<order_by_clause> ::=    ORDER BY { <order_by_expression> [, order_by_expression [ ,… ] ] }  
<order_by_expression> ::=  ... | DESC ] [ NULLS FIRST | NULLS LAST ]     | <position> [ <collate_clause> ] [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] }

<collation_name> is one of the supported collation names listed in the COLLATIONS system view.

collate_clause

Specifies the collation to use. <collate_clause> can only be used on columns defined as
NVARCHAR or VARCHAR and applies at the column level.

<collate_clause> ::= COLLABTE <collation_name> }

For example, in the following SELECT statement, collation ENGLISH only applies to column a, while
KOREAN only applies to column b.

SELECT * FROM t3 ORDER BY a collate ENGLISH, b collate KOREAN;

[ ASC | DESC ]

ASC sorts records in ascending order and DESC sorts records in descending order. The default is ASC.

[ NULLS FIRST | NULLS LAST ]

Specifies the ordering of NULL values.

position

<position> uses the entries in the select list to define the ordering required. It is an
<unsigned_integer>. For example:

SELECT col1, col2 FROM t ORDER BY 2;

ORDER BY 2 indicates that ordering should be undertaken using the second expression in the select
list, which in this case is col2.
limit_clause

Limits the number of records returned and behaves like TOP.

\[ <\text{limit clause}> ::= \text{LIMIT} <\text{unsigned integer}> \ [ \text{OFFSET} <\text{unsigned integer}> ] \ [ \text{TOTAL ROWCOUNT} ] \]

To ensure consistent, predictable results from the LIMIT clause, use the ORDER BY clause to sort the records in a specific order. Without an ORDER BY clause, the order of records is not guaranteed and the results of the LIMIT clause may change.

Specify TOTAL ROWCOUNT to update the session variable TOTAL_ROWCOUNT with the total number of rows the query would return if not limited. You can then query the value of the TOTAL_ROWCOUNT session variable (for example, `SELECT SESSION_CONTEXT('TOTAL_ROWCOUNT') FROM DUMMY;`).

Description

SELECT statements can be nested inside of statements such as another SELECT statement or an INSERT INTO or UPDATE statement. A SELECT statement that is nested in another statement is called a subquery. A subquery acts as a sort of filter, returning a set of data to the main statement that has some condition applied. For example, in the statement `SELECT last_name FROM Students WHERE student_id IN (SELECT student_id FROM Class_Registration),` the SELECT statement that is inside of the parenthesis is a subquery.

The subquery queries the Student_Registration table and returns (only) the IDs of students who have registered for a class. The outer, main query then queries the Students table for the last names that correspond with the student_ids retrieved from the Class_Registration table (this simplistic example presumes there is a student_ID column in both tables).

Privileges

You must have SELECT privileges on the object you are querying.

To query a system-versioned table, you must have the SELECT privilege on the table and on its associated history table.

Examples

Examples with WAIT and NOWAIT

```
CREATE COLUMN TABLE x ( a INT, b INT );
INSERT INTO x VALUES (1,1);
INSERT INTO x VALUES (2,2);
CREATE COLUMN TABLE y ( a INT, b INT );
INSERT INTO y VALUES (1,1);
INSERT INTO y VALUES (2,2);
SELECT * FROM x WHERE a=1 FOR UPDATE WAIT 1; --> OK
```
Examples with TIME TRAVEL

CREATE HISTORY COLUMN TABLE x ( a INT, b INT ); // after turning off auto commit
INSERT INTO x VALUES (1,1);
COMMIT;
SELECT last_commit_id FROM m_history_index_last_commit_id WHERE session_id = current_connection; // for example, 10
SELECT last_commit_id FROM m_history_index_last_commit_id WHERE session_id = current_connection; // for example, 20
DELETE FROM x;
COMMIT;
SELECT last_commit_id FROM m_history_index_last_commit_id WHERE session_id = current_connection; // for example, 30
SELECT * FROM x AS OF COMMIT ID 30; // return nothing
SELECT * FROM x AS OF COMMIT ID 20; // return two records (1,1) and (2,2)
SELECT * FROM x AS OF COMMIT ID 10; // return one record (1,1)
SELECT commit_time FROM sys.transaction_history WHERE commit_id = 10; // for example, '2012-01-01 01:11:11'
SELECT commit_time FROM sys.transaction_history WHERE commit_id = 20; // for example, '2012-01-01 02:22:22'
SELECT commit_time FROM sys.transaction_history WHERE commit_id = 30; // for example, '2012-01-01 03:33:33'
SELECT * FROM x AS OF UTCTIMESTAMP '2012-01-02 02:00:00'; // return one record (1,1)
SELECT * FROM x AS OF UTCTIMESTAMP '2012-01-03 03:00:00'; // return two records (1,1) and (2,2)
SELECT * FROM x AS OF UTCTIMESTAMP '2012-01-04 04:00:00'; // return nothing

Examples with WITH

CREATE TABLE t1(a INT, b INT);
WITH q1 AS (SELECT * FROM t1) SELECT * FROM q1;
WITH q1(c1, c2) AS (SELECT * FROM t1) SELECT * FROM q1;
WITH q1(c1) AS (SELECT a+b FROM t1), q2(c2) AS (SELECT MAX(c1) FROM q1)
SELECT c1 FROM q1 UNION ALL SELECT c2 FROM q2;

Example with FOR JSON

The following fictitious example shows what the results might look like when specifying FOR JSON with the options:

CREATE TABLE JTable (id int, name nvarchar(20));
INSERT INTO JTable VALUES (1, NULL);
INSERT INTO JTable VALUES (2, 'ko');

Normal operation:

SELECT id, name from JTable for JSON;
Returns:

```
[ {"id":1}, {"id":2, "name":"ko"} ]
```

Use of format = yes

```
SELECT id, name from JTable for JSON ('format'='yes');
```

Returns:

```
[{
  "ID": 1
}, {
  "ID": 2,
  "NAME": "ko"
}]
```

Use of the omitnull = no:

```
SELECT id, name from JTable for JSON ('format'='yes', 'omitnull'='no');
```

Returns:

```
[{
  "ID": 1,
  "NAME": null
}, {
  "ID": 2,
  "NAME": "ko"
}]
```

Use of arraywrap= no:

```
SELECT id, name from JTable for JSON ('format'='yes', 'omitnull'='no',
arraywrap'='no');
```

Returns:

```
{
  "ID": 1,
  "NAME": null
}, {
  "ID": 2,
  "NAME": "ko"
}
```

Example with FOR XML

The following fictitious example shows what the results might look like when specifying FOR XML with the schemaloc, targetns, and nullstyle options:

```
SELECT C1, C2 FROM T1
FOR XML ('schemaloc'='http://thiscompany.com/schemalib', 'targetns'='http://thiscompany.com/samples', 'nullstyle'='omit');
```
Examples with GROUP BY GROUPING SETS

Create a column table, t1, and populate it with some example data.

```
CREATE COLUMN TABLE t1 ( id INT PRIMARY KEY, customer VARCHAR(5), year INT, product VARCHAR(5), sales INT );
INSERT INTO t1 VALUES(1, 'C1', 2009, 'P1', 100);
INSERT INTO t1 VALUES(2, 'C1', 2009, 'P2', 200);
INSERT INTO t1 VALUES(3, 'C1', 2010, 'P1', 50);
INSERT INTO t1 VALUES(4, 'C1', 2010, 'P2', 150);
INSERT INTO t1 VALUES(5, 'C2', 2009, 'P1', 200);
INSERT INTO t1 VALUES(6, 'C2', 2009, 'P2', 300);
INSERT INTO t1 VALUES(7, 'C2', 2010, 'P1', 100);  INSERT INTO t1 VALUES(8, 'C2', 2010, 'P2', 150);
```

Use grouping sets to analyze the customer data.

```
SELECT customer, year, product, SUM(sales)      FROM t1
GROUP BY GROUPING SETS
( (customer, year),
  (customer, product) );
SELECT customer, year, NULL, SUM(sales)
FROM t1
GROUP BY customer, year
UNION ALL
SELECT customer, NULL, product, SUM(sales)
FROM t1
GROUP BY customer, product;
```

Observe that the two groups inside grouping sets in the first query are specified in each GROUP BY clause in the second query.

Use ROLLUP to generates results with multiple levels of aggregation:

```
SELECT customer, year, SUM(sales)      FROM t1
GROUP BY ROLLUP(customer, year);
SELECT customer, year, SUM(sales)      FROM t1
GROUP BY GROUPING SETS
( (customer, year),
  (customer) );
```
UNION ALL
SELECT NULL, NULL, SUM(sales)
FROM t1;

Use CUBE to generates results with multiple levels of aggregation.

SELECT customer, year, SUM(sales)
FROM t1
GROUP BY CUBE(customer, year);
SELECT customer, year, SUM(sales)
FROM t1
GROUP BY GROUPING SETS
{
  (customer, year),
  (customer),
  (year)
}
UNION ALL
SELECT NULL, NULL, SUM(sales)
FROM t1;

Use BEST 1 to return only the best group of results.

SELECT customer, year, product, SUM(sales)
FROM t1
GROUP BY GROUPING SETS BEST 1
{
  (customer, year),
  (product)
};

Use LIMIT 2 to limit the number of records to a maximum 2 for each group.

SELECT customer, year, product, SUM(sales)
FROM t1
GROUP BY GROUPING SETS LIMIT 2
{
  (customer, year),
  (product)
};

For the (customer, year) group, the number of records are 4, so only first 2 records are returned. For the (product) group, the number of records are 2, in this case all the records are returned.

Use WITH SUBTOTAL to produce an additional record for each group that displays a subtotal of returned records. These subtotal records are NULL for the customer, year, product columns and the sum of sum(sales) values in the select list.

SELECT customer, year, product, SUM(sales)
FROM t1
GROUP BY GROUPING SETS LIMIT 2 WITH SUBTOTAL
{
  (customer, year),
  (product)
};

Use WITH BALANCE to produce an additional record for each group that displays a subtotal of unreturned records.

SELECT customer, year, product, SUM(sales)
FROM t1
GROUP BY GROUPING SETS LIMIT 2 WITH BALANCE
Use WITH TOTAL to produce an additional record for each group that displays a total of all grouped records.

```sql
SELECT customer, year, product, SUM(sales)
FROM t1
GROUP BY GROUPING SETS LIMIT 2 WITH TOTAL
(
  (customer, year),
  (product)
);
```

Use TEXT_FILTER to retrieve the first column of each group with a given `<filterspec>`. The following query searches for columns ending with '2'. This is applied to customers in the first grouping set and products in the second. Only three matched records are returned.

```sql
SELECT customer, year, product, SUM(sales), TEXT_FILTER(customer),
TEXT_FILTER(product)
FROM t1
GROUP BY GROUPING SETS TEXT_FILTER '*2'
(
  (customer, year),
  (product)
);
```

Use FILL UP to return both matched and non-matched records with `<filterspec>`. The following query returns six records whereas the previous example only returned three.

```sql
SELECT customer, year, product, SUM(sales), TEXT_FILTER(customer),
TEXT_FILTER(product)
FROM t1
GROUP BY GROUPING SETS TEXT_FILTER '*2' FILL UP
(
  (customer, year),
  (product)
);
```

SORT MATCHES TO TOP is used to raise matched records up. For each grouping set, its grouped records are sorted.

```sql
SELECT customer, year, product, SUM(sales), TEXT_FILTER(customer),
TEXT_FILTER(product)
FROM t1
GROUP BY GROUPING SETS TEXT_FILTER '*2' FILL UP SORT MATCHES TO TOP
(
  (customer, year),
  (product)
);
```

Use STRUCTURED RESULT to create temporary tables, one for each grouping set and an additional table for the overview table.

```sql
SELECT customer, year, product, SUM(sales)
FROM t1
GROUP BY GROUPING SETS STRUCTURED RESULT
(
  (customer, year),
```
"#GN1" table is for (customer, year) grouping set and "#GN2" table is for (product) one.

Each table contains only related columns. That is to say, "#GN1" table does not have "product" column and "#GN2" table does not have "customer" and "year" columns.

Use WITH OVERVIEW to create a temporary table "#GN0" for the overview table.

```
DROP TABLE "#G1";
DROP TABLE "#G2";
SELECT customer, year, product, SUM(sales)
FROM t1
GROUP BY GROUPING SETS STRUCTURED RESULT WITH OVERVIEW
(
  (customer, year),
  (product)
);
```

Change the names of temporary tables by using the PREFIX keyword.

```
SELECT customer, year, product, SUM(sales)
FROM t1
GROUP BY GROUPING SETS STRUCTURED RESULT WITH OVERVIEW PREFIX '#MYTAB'
(
  (customer, year),
  (product)
);
```

Temporary tables are dropped when the corresponding session is closed or when a user executes a drop command. A list of temporary tables is seen in m_temporary_tables.

```
SELECT * FROM m_temporary_tables;
```

Use MULTIPLE RESULTSETS to return multiple result sets. The following query returns three result sets, one for the overview table and two for the grouping sets.

```
SELECT customer, year, product, SUM(sales)
FROM t1
GROUP BY GROUPING SETS MULTIPLE RESULTSETS
(
  (customer, year),
  (product)
);
```

**Example with TABLESAMPLE**

Use TABLESAMPLE to select a random sample of 1% of the employee table and within that sample count the number of managers.

```
SELECT COUNT(*), AVG(salary)
```
FROM employee TABLESAMPLE SYSTEM (1)     WHERE employee.type = 'manager';

Examples with EXPRESSION MACROS

The following example creates a view with an expression macro, avgA, and then selects from that view.

```
CREATE TABLE t1(a INT);
INSERT INTO t1 VALUES(10);
INSERT INTO t1 VALUES(20);
CREATE VIEW v1 AS SELECT * FROM t1 WITH EXPRESSION MACROS(AVG(a) AS avgA);
SELECT EXPRESSION_MACRO(avgA) FROM v1;
```

The following example creates a view, v1, with expression macros, sum_x and count_y:

```
CREATE TABLE t (x INT, y INT);
INSERT INTO t VALUES(1, 1);
INSERT INTO t VALUES(2, 1);
INSERT INTO t VALUES(3, 1);
INSERT INTO t VALUES(4, 1);
CREATE VIEW V1 AS SELECT * FROM t WITH EXPRESSION MACROS (     SUM(x) AS sum_x,
     COUNT(y) AS count_y   );
```

The following statements create another view, v2, that uses expression macros from the first view and gives them the new aliases, sum_x2 and count_y2.

```
CREATE VIEW v2 AS SELECT * FROM v1 WITH EXPRESSION MACROS (     sum_x AS sum_x2,
     count_y AS count_y2   );
```

The following statement selects the expression macro values from v2. The final SELECT statement returns 10, 4.

```
SELECT EXPRESSION_MACRO(sum_x2), EXPRESSION_MACRO(count_y2) FROM v2;
```

Example with HINTS

In the following example the USE_OLAP_PLAN hint is applied.

```
SELECT T2.* FROM ( SELECT MAX(COL) FROM T1 GROUP BY COL WITH
 HINT( NO_USE_OLAP_PLAN ) ) T2 WITH HINT( USE_OLAP_PLAN );
```

In the following example no hint is applied.

```
SELECT T2.* FROM ( SELECT MAX(COL) FROM T1 GROUP BY COL WITH
 HINT( NO_USE_OLAP_PLAN ) ) T2;
```

Examples for cache controlling hints:

```
SELECT * FROM T1 WITH HINT( IGNORE_PLAN_CACHE );
SELECT * FROM T1 WITH HINT( USE_REMOTE_CACHE );
```

Examples for the use of hints in a scale-out environment:

```
SELECT * FROM T1 WITH HINT( ROUTE_TO(1));
SELECT * FROM T1 WITH HINT( NO_ROUTE_TO(2,3));
SELECT * FROM T1 WITH HINT( ROUTE_BY(T2));
```
Examples for execution engine selection with hints:

```
SELECT * FROM T1 WITH HINT( USE_OLAP_PLAN );
SELECT * FROM T1 WITH HINT( NO_USE_OLAP_PLAN );
```

Examples for controlling the access path with hints:

```
SELECT * FROM T1 WITH HINT( INDEX_SEARCH );
SELECT * FROM T1 WITH HINT( NO_INDEX_SEARCH );
```

Examples for controlling join operations with hints:

```
SELECT * FROM T1, T2 WITH HINT( INDEX_JOIN );
SELECT * FROM T1, T2 WITH HINT( NO_INDEX_JOIN );
SELECT * FROM T1, T2 WITH HINT( HASH_JOIN );
SELECT * FROM T1, T2 WITH HINT( NO_HASH_JOIN );
SELECT * FROM T1, T2 WITH HINT( MIXED_INVERTED_INDEX_JOIN );
SELECT * FROM T1, T2 WITH HINT( NO_MIXED_INVERTED_INDEX_JOIN );
SELECT * FROM T1, T2 WITH HINT( OPTIMIZE_METAMODEL );
SELECT * FROM T1, T2 WITH HINT( NO_OPTIMIZE_METAMODEL );
```

Examples for query rewriting and logical transformations with hints:

```
SELECT * FROM T1 WITH HINT( SUBPLAN_SHARING);
SELECT * FROM T1 WITH HINT( NO_SUBPLAN_SHARING );
```

Examples of selecting from arrays

Select values from the array column C1 in table T1.

```
SELECT C1 FROM T1;
```

Select an array value using array construction by enumeration.

```
SELECT ARRAY ( 1, 2, 3, 4 ) FROM DUMMY;
```

Select an array value using array construction by query.

```
SELECT ARRAY( SELECT C1 FROM T0 ) FROM DUMMY;
```

Example with PARTITION

You use PARTITION to select the data contained in partitions 1, 3 and 4 of table T0.

```
SELECT * FROM T0 PARTITION (1, 3, 4);
```

Example with FOR SHARE LOCK

This example involves 3 sessions and all of them are in auto-commit-off mode and read-committed isolation level:

```
-- in Session 1:
CREATE TABLE X (A INT);
INSERT INTO X VALUES (0);
COMMIT;
-- in session 2:
```
SELECT A FROM X FOR SHARE LOCK;    -- OK, {0} returned
-- in session 3:
SELECT A FROM X FOR SHARE LOCK;    -- OK, {0} returned. Now, record 0 has two
shared-lock owners: session 2 and Session 3
-- in session 1:
SELECT A FROM X;                   -- OK, {0} returned; read access is allowed
UPDATE X SET A = 1;                -- lock wait

In the above example, the UPDATE statement tries to acquire exclusive-lock on record 0. If session 1 and
session 3 issue either a COMMIT or a ROLLBACK before session 1’s lock timeout is reached, then session
1 will successfully update the record. If session 2 or session 3 holds the lock longer than session 1’s lock
timeout, then session 1 will get a lock timeout error and will be rolled back.

The next example involves two sessions, both of which are in auto-commit-off mode and read-committed
isolation level, and show how FOR SHARE LOCK guarantees that there are no on-going writers on the
queried records.

--session 1:
CREATE TABLE X (A INT);
INSERT INTO X VALUES (0);
COMMIT;
UPDATE X SET A = 1;               -- OK, updated records = 1
--session 2:
SELECT A FROM X FOR SHARE LOCK;   -- lock wait, since session 1 is holding
exclusive lock on record 0 (for a maximum of session 2’s lock timeout value)
--session 1:
COMMIT;                           -- now that session 1 has executed a COMMIT
statement, session 2 can proceed and a 1 is returned.

Note that session 1 is in read-committed isolation level; thus, the updated record is returned after waiting
for the record lock.

This example shows a SHARE LOCK OF with a subset of a primary key.

-- Create table:
CREATE TABLE TAB1 (A INT, B INT, C NVARCHAR(10), PRIMARY KEY(A, B));
INSERT INTO TAB1 VALUES (1, 1, 'insert');
COMMIT;
-- Transaction 1 acquires key shared lock (autocommit=off):
SELECT * FROM TAB1 FOR SHARE LOCK OF A;
-- Transaction 2 can update the same record, when the key is not modified
(not blocked):
UPDATE TAB1 SET C = 'update' WHERE A = 1 AND B = 1;
-- For transaction 2, the change of key is still blocked as before:
UPDATE TAB1 SET A = 2 WHERE A = 1 AND B = 1;

This example shows a SHARE LOCK OF with a non-key column.

-- Create table:
CREATE TABLE TAB2 (A INT, B INT, V NVARCHAR(10), PRIMARY KEY(A,Bb));
INSERT INTO TAB2 VALUES (1, 1, 'insert');
COMMIT;
-- Transaction 1 acquires key shared lock (autocommit=off):
SELECT * FROM TAB2 FOR SHARE LOCK OF A, C;
-- Transaction 2 is blocked by transaction 1, even though key is unmodified:
UPDATE TAB2 SET C = 'update' WHERE A = 1 AND B = 1;

Examples with ASSOCIATIONS
The following statements create three tables and populates them with data:

```
CREATE ROW TABLE EMPLOYEES (ID INT, NAME VARCHAR(20), CITY_ID INT) WITH ASSOCIATIONS (JOIN CITIES AS CITY ON CITY.ID = CITY_ID);
CREATE ROW TABLE CITIES (ID INT, NAME VARCHAR(20), STATE_ID INT) WITH ASSOCIATIONS (JOIN STATES AS STATE ON STATE.ID = STATE_ID);
CREATE ROW TABLE STATES (ID INT, NAME VARCHAR(20));
INSERT INTO STATES VALUES(1, 'California');
INSERT INTO STATES VALUES(2, 'Washington');
INSERT INTO CITIES VALUES(1, 'San Francisco', 1);
INSERT INTO CITIES VALUES(2, 'Los Angeles', 1);
INSERT INTO CITIES VALUES(3, 'Oakland', 1);
INSERT INTO EMPLOYEES VALUES(1, 'Tom', 1);
INSERT INTO EMPLOYEES VALUES(2, 'Jerry', 2);
INSERT INTO EMPLOYEES VALUES(3, 'Mick', 1);
INSERT INTO EMPLOYEES VALUES(4, 'Steve', 3);
```

The following statement queries the EMPLOYEES table using the association with the CITIES table:

```
SELECT * FROM EMPLOYEES:CITY.STATE;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>California</td>
</tr>
<tr>
<td>2</td>
<td>Washington</td>
</tr>
</tbody>
</table>

The following statement filters values in the ID columns of the EMPLOYEES and CITIES tables in the query:

```
SELECT * FROM EMPLOYEES[ID<5]:CITY.STATE[ID<2];
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>California</td>
</tr>
</tbody>
</table>

**Example with CASE JOIN**

The following example shows how to use a CASE JOIN to choose a different table to be joined for each record that matches:

```
CREATE COLUMN TABLE t1 (branch INT, id INT, c1 INT, c2 INT);
CREATE COLUMN TABLE t2 (id INT, c1 INT, c2 INT);
CREATE COLUMN TABLE t3 (id INT, c1 INT, c2 INT);
INSERT INTO t1 VALUES (1, 1, 100, 200);
INSERT INTO t1 VALUES (2, 1, 10, 20);
INSERT INTO t2 VALUES (1, 100, 200);
INSERT INTO t3 VALUES (1, 300, 400);

SELECT t1.branch, case_join.c1, case_join.c2
FROM t1 LEFT OUTER MANY TO ONE CASE JOIN
WHEN t1.branch = 1 THEN RETURN (c1, c2) FROM t2 ON t1.id = t2.id
WHEN t1.branch = 2 THEN RETURN (c1, c2) FROM t3 ON t1.id = t3.id
END AS case_join;
```
Example using TOTAL_ROWCOUNT to determine total number of rows that would be returned

The following example shows how to use the TOTAL_ROWCOUNT session variable with LIMIT to determine the number of rows a query would return, without having to fetch all of the rows:

```
CREATE COLUMN TABLE test_rowcount(a INT, b INT);
INSERT INTO test_rowcount VALUES (1,1);
INSERT INTO test_rowcount VALUES (2,2);
INSERT INTO test_rowcount VALUES (3,3);
SELECT * FROM test_rowcount LIMIT 1 TOTAL ROWCOUNT; <-- TOTAL_ROWCOUNT session variable is updated to 3
SELECT SESSION_CONTEXT('TOTAL_ROWCOUNT') FROM DUMMY; <-- Returns the value of the session variable (3)
```

Related Information

User-Defined Functions
Declarative SQLScript Logic
Table Variable Type Definition
EXPRESSION_MACRO Function (Miscellaneous) [page 193]
Expressions [page 56]
COLLABORS System View [page 1520]
HINTS System View [page 1585]
HINT Details [page 1134]
M_CS_PARTITIONS System View - Deprecated [page 1833]
TRANSACTION_HISTORY System View [page 1693]
M_HISTORY_INDEX_LAST_COMMIT_ID System View [page 1927]
Introduction to SQL [page 27]
GROUPING_ID Function (Miscellaneous) [page 204]
GROUPING Function (Miscellaneous) [page 202]
https://www.w3schools.com/xml/xml_elements.asp

4.10.1.152.1 HINT Details

The SQL Optimizer usually determines the access path (for example, index search versus table scan) on the basis of the costs (Cost-Based Optimizer). You can override the SQL Optimizer choice by explicitly specifying hints in the query that enforces a certain access path.
Hint Syntax in SQL Statements

```
{ SELECT | INSERT | UPDATE | DELETE } ... <hint_clause>
<hint_clause> ::= WITH HINT( <hint> [, <hint> ...])
<hint> ::= { <hint_element> | <routing_hint> | <value_input> } [ CASCADE ]
<hint_element> ::= <hint_name> from public hint list [{(<hint_argument_list>)}]
<hint_argument_list> ::= <hint_argument> [, <hint_argument> ...]
<hint_argument> ::= hint argument type according to each hint specification (for example, one of STR_CONST, INT_CONST, ID, or <table_ref>)

<routing_hint> ::= ROUTE_TO( <volumd_id> [, <volumd_id> [, ...]])
| ROUTE_TO( <service_type> [, <service_type> [, ...]])
| NO_ROUTE_TO( <volumd_id> [, <volumd_id> [, ...]])
| ROUTE_BY( { <table_name> | <projection_view_name> } [, { <table_name> | <projection_view_name> } [, ...]])
| ROUTE_BY_CARDINALITY( { <table_name> | <projection_view_name> } [, { <table_name> | <projection_view_name> } [, ...]])

=value_input> ::= MAX_CONCURRENCY (1) | DATA_TRANSFER_COST (0 | 1)
```

When there are hint clauses in the subquery, only the outermost statement’s hint is applied. Additionally, hint arguments with a framework are also applied.

In the following example, the USE_OLAP_PLAN hint is applied:

```
SELECT T2.* FROM (SELECT MAX(COL) FROM T1 GROUP BY COL WITH HINT( NO_USE_OLAP_PLAN )) T2 WITH HINT( USE_OLAP_PLAN );
```

In the following example, no hint is applied because the hint clause is not part of the outermost statement:

```
SELECT T2.* FROM (SELECT MAX(COL) FROM T1 GROUP BY COL WITH HINT( NO_USE_OLAP_PLAN )) T2;
```

Hint Validation

Prior to SPS 09 incorrect syntax and/or semantics generates an SQL warning.
Starting from SPS 09 incorrect syntax and/or semantics generates an SQL error.

Hints by Category

Refer to each category for a more detailed description.
Hints for Controlling Request Processing

**THROW_ERROR**

Blocks the execution of a given query.

Examples:

```
SELECT * FROM T1 WITH HINT( THROW_ERROR );
```

```
ALTER SYSTEM ADD STATEMENT HINT ( THROW_ERROR ) FOR SELECT * FROM T1;
```

Hints for Controlling Cache Behaviors

SAP HANA supports different cache types.

The SQL Plan Cache saves compiled plans so that the same query does not need to be compiled every time. This setting saves CPU and compilation time and enables the gathering of statistics (user name, execution count, avg execution time, and so on) for each plan.

**IGNORE_PLAN_CACHE**

Ignores the existing plan cache entry (if any) and forces the query to compile and execute.

- If a plan cache entry already exists then it is ignored but still remains in the plan cache.
- If a plan cache entry does not exist, then no plan cache entry is added.

**USE_REMOTE_CACHE**

Optimizes SAP HANA Hadoop and forces virtual table queries to have to a materialized result set. Subsequent queries are served from the materialized view.

Examples:

```
SELECT * FROM T1 WITH HINT( IGNORE_PLAN_CACHE );
```

```
SELECT * FROM T1 WITH HINT( USE_REMOTE_CACHE );
```

Hints for Scale-out Environment

For a scale-out environment, the SQL optimizer decides on the execution location for each operator based on the operator cost and the data transfer cost. Query routing is determined in a round-robin manner to one of the nodes that contains related objects, such as tables and views.

You may want to specify preferred nodes or objects for routing for performance or any other possible issues. Volume ID or table name can be supplied to hints to enable such routing.

**Note**

Wrong volume IDs generate an SQL warning.
**i Note**

If more than one hint is listed or nested, then the order of parsing determines which hint to apply. The hint that was given last takes effect.

---

**ROUTE_TO**

Routes the query to the specified volume ID or service type.

**NO_ROUTE_TO**

Avoids query routing to a specified volume ID or service type.

**ROUTE_BY**

Routes the query to the hosts related to the base table(s) of the specified projection view(s).

**ROUTE_BY_CARDINALITY**

Routes the query to the hosts related to the base table(s) of the specified projection view(s) with the highest cardinality from the input list.

**DATA_TRANSFER_COST (value)**

Guides the optimizer to use the weighting factor for the data transfer cost. The value 0 ignores the data transfer cost.

**ROUTE_OPTIMIZATION_LEVEL (MINIMAL | ALL)**

Guides the optimizer to compile with ROUTE_OPTIMIZATION_LEVEL (MINIMAL) or to default to ROUTE_OPTIMIZATION_LEVEL. If the MINIMAL compiled plan is cached, then it compiles once more using the default optimization level during the first execution. This hint is primarily used to shorten statement routing decisions during the initial compilation.

Example:

```sql
SELECT * FROM T1 WITH HINT (DATA_TRANSFER_COST(0));
```

The following example routes to the node where the volume ID = 1:

```sql
CALL P1() WITH HINT( ROUTE_TO(1));
```

The following example routes to one of the nodes with a volume ID other than 2 or 3:

```sql
CALL P1() WITH HINT( NO_ROUTE_TO(2,3));
```

The following example routes to the node containing table T2:

```sql
CALL P1() WITH HINT( ROUTE_BY(T2));
```

The following example routes to the node that has the highest cardinality among tables T1, T2, or T3:

```sql
CALL P1() WITH HINT( ROUTE_BY_CARDINALITY(T1,T2,T3));
```

The following example routes to a volume ID = 1 (where the last 1 is used):

```sql
CALL P1() WITH HINT( NO_ROUTE_TO(1), ROUTE_TO(1));
```
Hints for OLAP Engine

The Column Search execution engine is chosen automatically by the optimizer based on the cost model.

- **USE_OLAP_PLAN**
  - Guides the optimizer to prefer the OLAP engine over other engines.
- **NO_USE_OLAP_PLAN**
  - Guides the optimizer to avoid the use of the OLAP engine.
- **OLAP_PARALLEL_AGGREGATION (deprecated)**
  - Guides the optimizer to prefer the OLAP engine for a column search.

Examples:

```sql
SELECT * FROM T1 WITH HINT( USE_OLAP_PLAN );

SELECT * FROM T1 WITH HINT( NO_USE_OLAP_PLAN );
```

Hints for ESX Engine Selection

Use the ESX engine to execute a query.

- **USE_ESX_PLAN**
  - Guides the optimizer to choose the ESX engine over other engines.
- **NO_USE_ESX_PLAN**
  - Guides the optimizer to avoid the use of the ESX engine.

Examples:

```sql
SELECT * FROM T1 WITH HINT( USE_ESX_PLAN );

SELECT * FROM T1 WITH HINT( NO_USE_ESX_PLAN );
```

Hints for HEX Engine Selection

Use the HEX engine to execute a query.

- **USE_HEX_PLAN**
  - Guides the optimizer to choose the HEX engine over other engines.
- **NO_USE_HEX_PLAN**
  - Guides the optimizer to avoid the use of the HEX engine.

Examples:

```sql
SELECT * FROM T1 WITH HINT( USE_HEX_PLAN );
```
Hints for HEX Enumeration Decisions

The following hints determine special hex enumeration decisions that have performance impacts if the query goes through HEX enumeration.

HEXHASHJOIN
Guides the optimizer to prefer HEX hash joins over other joins.

NO_HEX_HASH_JOIN
Guides the optimizer to avoid HEX hash joins.

HEXINDEXJOIN
Guides the optimizer to prefer HEX index joins over other joins.

NO_HEX_INDEX_JOIN
Guides the optimizer to avoid HEX index joins.

HEXNESTEDLOOPJOIN
Guides the optimizer to prefer HEX nested loop joins over other joins.

NO_HEX_NESTED_LOOP_JOIN
Guides the optimizer to avoid HEX nested loop joins.

CONCAT_FILTER
Guides the optimizer to prefer HEX concat replacements.

NO_CONCAT_FILTER
Guides the optimizer to avoid HEX concat replacements.

HEXRANGEJOIN
Guides the optimizer to prefer HEX range joins over other joins.

NO_HEX_RANGE_JOIN
Guides the optimizer to avoid HEX range joins.

HEXHASHEDRANGEJOIN
Guides the optimizer to prefer HEX hashed range joins over other joins.

NO_HEX_HASHED_RANGE_JOIN
Guides the optimizer to avoid HEX hashed range joins.

HEXTABLESCAN
Guides the optimizer to prefer HEX table scans over unique index searches.

NO_HEX_TABLE_SCAN
Guides the optimizer to avoid HEX table scans.

HEXUNIQUE_INDEX_SEARCH
Guides the optimizer to prefer HEX unique index searches over table scans.

NO_HEX_UNIQUE_INDEX_SEARCH
Guides the optimizer to avoid HEX unique index searches.

**HEX_LIMIT**

Guides the optimizer to prefer HEX limits over top K sorts.

**NO_HEX_LIMIT**

Guides the optimizer to avoid HEX limits.

**HEX_TOPK_SORT**

Guides the optimizer to prefer HEX top K sorts over HEX limits.

**NO_HEX_TOPK_SORT**

Guides the optimizer to avoid HEX top K sorts.

### Hints for Join Engine Optimization

The following hints force a certain join operation behavior:

**OPTIMIZE_METAMODEL**

Creates concat attributes for joins with multiple columns (if they do not already exist) and uses them during the join calculation. Concat attributes need additional memory as they are persisted.

**NO_OPTIMIZE_METAMODEL**

Forces the use of the native hash-based multi-column join implementation and does not use concat attributes even if they should have already been created.

**MAX_CONCURRENCY (1)**

Controls concurrency. This setting only accepts the value 1 (single thread plan execution) and the Join Engine determines suitable parallelism by default.

**CS_JOIN_RESULT_MATERIALIZATION( [ schema_name : ] table_name )**

Specifies the table at which join materialization begins.

**CS_JOIN_REDUCTION( [ schema_name : ] table_name )**

Specifies the table at which the reduction phase begins.

**CS_JOIN_CYCLE_BREAK( [ schema_name : ] table_name_left , [ schema_name : ] table_name_right )**

Specifies the join edge at which a cyclic join is broken up.

**CS_JOIN_MATERIALIZATION_BIGGEST_TABLE_FIRST**

Examples:

```sql
SELECT * FROM T1 WITH HINT( OPTIMIZE_METAMODEL );

SELECT * FROM T1 WITH HINT( NO_OPTIMIZE_METAMODEL );

SELECT * FROM T1 WITH HINT( MAX_CONCURRENCY(1) );
```
Specifies that the materialization starts at the biggest table.

CS_JOIN_MATERIALIZATION_SMALLEST_TABLE_FIRST

Specifies that the materialization starts at the smallest table.

CS_JOIN_MATERIALIZATION_CONCURRENCY(number)

Defines the concurrency of the join materialization.

CS_JOIN_SHRINK_SIZE_THRESHOLD(number)

Specifies that the intermediate result exceeds this threshold, then tries to shrink it.

CS_JOIN_POST_CHECK_SIZE_THRESHOLD(number)

Specifies the intermediate result exceeds this threshold, then runs post-conditions and distinct checks on it.

CS_JOIN_REDUCTION_MULTICOLUMN_FIRST(table_namecolumn_name)

Specifies the column to be evaluated first in the JoinEngine semi-join reduction, in the case of a multicolumn join with skewed data distribution.

CS_JOIN_FILTER_AFTER_REDUCTION( '[ schema_name :] table_name ')

Applies filters on a table after a join reduction in the join engine.

CS_JOIN_FILTER_BEFORE_REDUCTION( '[ schema_name :] table_name ')

Applies filters on a table before a join reduction in the join engine.

**Hints for Controlling Access Path**

The following hints override the cost-based decision of the optimizer in order to utilize or avoid indexes for table access.

**INDEX_SEARCH**

Guides the optimizer to access the table by using the available index.

**NO_INDEX_SEARCH**

Guides the optimizer to avoid table access by using the available index.

Examples:

```sql
SELECT * FROM T1 WITH HINT( INDEX_SEARCH );
```

```sql
SELECT * FROM T1 WITH HINT( NO_INDEX_SEARCH );
```

**Hints for Controlling Join Operations**

There are different join algorithms for performing join operations.

**INDEX_JOIN**

Guides the optimizer to join input relations through index searches.

**NO_INDEX_JOIN**
Guides the optimizer to avoid joining the input relations through index searches.

**NO_INDEX_JOIN**

Guides the optimizer to join the input relations through probing the hash table.

**NO_HASH_JOIN**

Guides the optimizer to avoid joining the input relations through probing the hash table.

**MIXED_INVERTED_INDEX_JOIN**

Guides the optimizer to join input relations of row store formats with a column table, without format conversion, using an inverted index of the column table.

**NO_MIXED_INVERTED_INDEX_JOIN**

Guides the optimizer to avoid joining input relations using an inverted index of the column table.

**OPTIMIZE_METAMODEL**

Guides the optimizer to avoid joining the input relations through probing the hash table.

Creates concat attributes for joins with multiple columns (if they do not already exist) and uses them during the join calculation. Concat attributes need additional memory as they are persisted.

**NO_OPTIMIZE_METAMODEL**

Forces the use of the native hash-based multi-column join implementation and does not use the concat attributes even if they should have already been created.

---

**Note**

The hash-based approach might be slower than using concat attributes.

---

**Examples:**

```sql
SELECT * FROM T1, T2 WITH HINT( INDEX_JOIN );
SELECT * FROM T1, T2 WITH HINT( NO_INDEX_JOIN );
SELECT * FROM T1, T2 WITH HINT( HASH_JOIN );
SELECT * FROM T1, T2 WITH HINT( NO_HASH_JOIN );
SELECT * FROM T1, T2 WITH HINT( OPTIMIZE_METAMODEL );
SELECT * FROM T1, T2 WITH HINT( NO_OPTIMIZE_METAMODEL );
```

---

**Hints for Controlling Row/Column Store Operations**

Most of the operations can be supported in both the row engine and the column engine. The optimizer selects the best operations based on the cost-model. The hints in this section can be used to control whether to prefer or avoid the use of specific engine operators in the query plan.

The following hints guide the optimizer to prefer the column engine operator:

- **CS_SORT**
The following hints guide the optimizer to prefer the row engine operator:

- NO_CS_SORT
- NO_CS_LIMIT
- NO_CS_FILTER
- NO_CS_DISTINCT
- NO_CS_AGGR
- NO_CS_JOIN
- NO_CS_UNION_ALL

Hints for Smart Data Access/Dynamic Tiering

Smart Data Access/Dynamic Tiering uses two kinds of operators to read data from the remote source: Remote Column Scan reads data from the remote server and stores it in ITAB and Remote Row Scan reads data from the remote server and stores it in a buffer.

- REMOTE_COLUMN_SCAN
  Guides the optimizer to prefer Remote Column Scan, which returns ITAB from the remote server.
- NO_REMOTE_COLUMN_SCAN
  Guides the optimizer to prefer Remote Row Scan.
- REMOTE_JOIN_RELOCATION
  Guides the optimizer to process the join at the remote server if possible (default behavior).
- NO_REMOTE_JOIN_RELOCATION
  Guides the optimizer to process the join at the HANA server.
- REMOTE_EXPR_MATERIALIZATION
  Guides the optimizer to prefer the expression evaluation at the remote server if possible (default behavior).
- NO_REMOTE_EXPR_MATERIALIZATION
  Guides the optimizer to prefer the expression evaluation at the HANA server if possible.
- REMOTE_PREAGGR
  Guides the optimizer to generate pre-aggregation at the remote server (default behavior).
- NO_REMOTE_PREAGGR

Note

A full list of hints to control other row engine or column engine operators can be found in the HINTS view. Using hints that are not described in this section is not recommended.
Guides the optimizer not to generate pre-aggregation at the remote server.

**REMOTE_AGGR** Guides the optimizer to process aggregation at remote data source if possible (default behavior).

**NO_REMOTE_AGGR** Guides the optimizer to process aggregation at HANA server if possible.

**REMOTE_JOIN** Guides the optimizer to process the join at the remote server if possible. When used with **NO_REMOTE_JOIN_RELOCATION**, it prefers processing of the join involving tables from the same remote source at the remote server (default behavior).

**NO_REMOTE_JOIN** Guides the optimizer to process the join between tables at HANA server, even if the tables are from the same remote source.

**REMOTE_DISTINCT** Guides the optimizer to process the distinct operation at the remote server if possible (default behavior).

**NO_REMOTE_DISTINCT** Guides the optimizer to process the distinct operation at the HANA server if possible.

**REMOTE_UNION** Guides the optimizer to process union operation at the remote server if possible (default behavior).

**NO_REMOTE_UNION** Guides the optimizer to process the union operation at the HANA server if possible.

Examples:

```sql
SELECT * FROM T1 WITH HINT( REMOTE_COLUMN_SCAN );
SELECT * FROM T1 WITH HINT( REMOTE_JOIN_RELOCATION );
SELECT * FROM T1 WITH HINT( REMOTE_EXPR_MATERIALIZATION );
SELECT * FROM T1 WITH HINT( REMOTE_PREAGGR );
SELECT SUM(HT.C1) FROM RT JOIN HT ON RT.C1 = HT.C1 GROUP BY RT.C1 WITH HINT(REMOTE_AGGR)
SELECT * FROM RT JOIN HT ON RT.C1 = HT.C1 WITH HINT(REMOTE_JOIN)
SELECT DISTINCT(HT.C1) FROM RT JOIN HT ON RT.C1 = HT.C1 WITH HINT(REMOTE_DISTINCT)
SELECT * FROM RT UNION SELECT * FROM HT WITH HINT(REMOTE_UNION)
```

**Hints for Result Cache**

Use the result cache for applications that accept stale data access. Table functions and SQL/Calculation views can be cached. Since the result cache shows stale data, it can only be used when suitable hints are turned on.

**RESULT_CACHE**

Guides the optimizer to utilize the result cache regardless of indexserver configuration, if the result cache is available.

**NO_RESULT_CACHE**

Guides the optimizer to not utilize the result cache, even if it is available.
RESULT_CACHE_MAX_LAG(seconds)
Guides the optimizer to set the retention period of the result cache to a minimum of this value or the value set in the ADD CACHE configuration.

RESULT_CACHE_NON TRANSACTIONAL
Allows join or union operations using the result cache entries and disregards possible transaction inconsistencies.

RESULT_CACHE_NO_REFRESH
Accesses cached data if cached data is prepared before, without refreshing it, even when its retention period is over.

RESULT_LAG(‘name’[, seconds])
Guides the optimizer to use specified engines if its result lag is less than the maximum seconds.

RESULT_CACHE_AFTER_ANALYTIC_PRIVILEGE
Enforces the result cache to use separate data cached after filtering with analytic privileges. This hint does not enforce using the result cache and it is only effective when the query can use the result cache.

RESULT_CACHE_BEFORE_ANALYTIC_PRIVILEGE
Enables the result cache to use shared data cached before filtering with analytic privileges. This hint does not enforce using the result cache and it is only effective when the query can use the result cache.

Examples:

```
SELECT * FROM T1 WITH HINT( NO_RESULT_CACHE );
SELECT * FROM T1 WITH HINT( RESULT_CACHE );
SELECT * FROM T1 WITH HINT( RESULT_CACHE_MAX_LAG(60) );
SELECT * FROM T1 WITH HINT( RESULT_CACHE_NON TRANSACTIONAL );
SELECT * FROM T1 WITH HINT( RESULT_LAG (’hana_long’) );
SELECT * FROM T1 WITH HINT( RESULT_LAG (’hana_long’, 30) );
SELECT * FROM T1 WITH HINT( RESULT_CACHE_AFTER_ANALYTIC_PRIVILEGE );
SELECT * FROM T1 WITH HINT( RESULT_CACHE_BEFORE_ANALYTIC_PRIVILEGE );
```

Hints for Dynamic Result Cache

Returns an up-to-date query result. This means that the result of the query run on the dynamic result cache is always the same as the result of a query that is not using the cache.

NO_DYNAMIC_RESULT_CACHE
Guides the optimizer not to utilize the dynamic result cache regardless of indexserver configuration, even if it is available.
**DYNAMIC_RESULT_CACHE**
Guides the optimizer to utilize the dynamic result cache if the dynamic result cache is available (default).

**NO_DYNAMIC_RESULT_CACHE**
Guides the optimizer not to try automatic view matching (default).

**DYNAMIC_RESULT_CACHE_IMPLICIT_MATCH**
Guides the optimizer to try automatic view matching regardless of indexserver configuration.

<table>
<thead>
<tr>
<th>HINT CALL</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DYNAMIC_RESULT_CACHE_IMPLICIT_MATCH(schema_name.view_name[,schema_name.view_name ...])</td>
<td>Guides the optimizer to try automatic view matching only with given views.</td>
</tr>
</tbody>
</table>

Examples:

```
SELECT * FROM V1 WITH HINT( NO_DYNAMIC_RESULT_CACHE );
SELECT * FROM V1 WITH HINT( DYNAMIC_RESULT_CACHE);  
SELECT * FROM T1 WITH HINT( NO_DYNAMIC_RESULT_CACHE_IMPLICIT_MATCH );
SELECT * FROM T1 WITH HINT( DYNAMIC_RESULT_CACHE_IMPLICIT_MATCH );
SELECT * FROM T1 WITH HINT( DYNAMIC_RESULT_CACHE_IMPLICIT_MATCH ("TEST"."V1") );
```

**Hints for Asynchronous Table Replication**

Used for applications that accept stale data access. Since the asynchronous table shows stale data, it can only be used with suitable hints or to access the replica table by using the explicit replica table name.

**RESULT_LAG('name' [,seconds])**
Guides the optimizer to access source or replica tables by evaluating stale data with the \( <\text{seconds}> \) parameter.

Examples:

```
SELECT * FROM T1 WITH HINT(RESULT_LAG ('hana_short')) ;
SELECT * FROM T1 WITH HINT(RESULT_LAG ('hana_atr')) ;
```

**Hints for Column View Cardinality Estimation**

Column Views (Join views, OLAP views, and Calculation views) are relations, which can be used in SQL plans. Cardinality estimation of such view objects is necessary for the SQL Optimizer to find an optimal plan.
The optimizer has three different estimation methods, each of which has its own trade-offs in terms of the quality and efficiency. The optimizer utilizes an appropriate method among them, considering the complexity of a given plan.

These hints maneuver the optimizer to use a designated method for cardinality estimation of all of the related Column views.

- **NO_COLUMN_VIEW_ESTIMATION**
  
  Gets a record count estimation of the largest base table (legacy).

- **COLUMN_VIEW_ESTIMATION(RECORD COUNT)**
  
  Guides the optimizer to use the output record count estimation, which is done in each engine layer.

- **COLUMN_VIEW_ESTIMATION(SIMPLE)**
  
  Guides the optimizer to use a comprehensive estimation method to retrieve SIMPLE histograms (record count, distinct value count, and so on) for all of the requested attributes.

- **COLUMN_VIEW_ESTIMATION**
  
  Guides the optimizer to utilize the latest estimation method.

- **COLUMN_VIEW_LARGE_PREDICATE_SAMPLING**
  
  Guides the optimizer to sample predicates for large predicates (greater than 200 terms) inside of a column view.

- **NO_COLUMN_VIEW_LARGE_PREDICATE_SAMPLING**
  
  Forbids the optimizer to sample predicates for large predicates (greater than 200 terms) inside of a column view.

Examples:

```sql
SELECT * FROM T1 WITH HINT( NO_COLUMN_VIEW_ESTIMATION );
SELECT * FROM T1 WITH HINT( COLUMN_VIEW_ESTIMATION (RECORD COUNT) );
SELECT * FROM T1 WITH HINT( COLUMN_VIEW_ESTIMATION (SIMPLE) );
SELECT * FROM T1 WITH HINT( COLUMN_VIEW_ESTIMATION );
```

### Hints for Access Path Optimization on Column Store Tables

- **CS_PRIMARY_KEY**
  
  Forces the use of the primary key, even though the optimizer estimates otherwise. This hint is only applicable if the key is not fully defined in the query.

- **NO_CS_PRIMARY_KEY**
  
  Forbids the use of the primary key, although the optimizer estimates otherwise. This hint is only applicable if the key is not fully defined in the query.

- **CS_ESTIMATION(SIMPLE | ALL)**
  
  Controls the estimation phase of the column-store table query. ALL (default) estimates all of the predicates and SIMPLE skips predicates that are too expensive to estimate.
**NO_CS_ESTIMATION**

Skips the estimation phase of the column-store table query completely.

**CS_FILTER_FIRST(column)**

Specifies the preferred starting column in a conjunction.

---

**Hints for Active/Active (Read Enabled)**

Routes the statement from the primary site to a secondary site on an Active/Active (read enabled) system.

**RESULT_LAG('hana_sr' [ , seconds ])**

Guides the optimizer to use the secondary system. The routed statement queries data on the secondary system. The lag time is the time between the commit on the primary system and visibility on the secondary system. If the lag time on the secondary system exceeds the specified time on the hint, then the statement may be re-routed to the primary system implicitly. For example:

Examples:

```
SELECT * FROM T1 WITH HINT( RESULT_LAG ('hana_sr') );
```
```
SELECT * FROM T1 WITH HINT( RESULT_LAG ('hana_sr', 60) );
```

---

**Hints for Workload Management**

Provides a method to set another workload class for a query, regardless of any workload mappings.

**WORKLOAD_CLASS(workload_class_name)**

Designates a certain workload class for the statement execution:

- The workload class must be created before using this hint.
- If the specified workload class does not exist, the hint is ignored and the corresponding SQL warning is returned for the statement execution.
- Only workload classes which have more restricted properties compared to the ones set for session-wise admission control are applied. Otherwise, the hint is ignored and the corresponding SQL warning is returned. This hint clause only allows priority, thread limit, and memory limit down-wise.

Examples:

```
CREATE WORKLOAD CLASS "MY_WORKLOAD_CLASS_1" SET 'PRIORITY'='9';
```
```
SELECT * FROM T1 WITH HINT( WORKLOAD_CLASS("MY_WORKLOAD_CLASS_1") );
```

The following examples suppose that two workload classes are matched to the current execution request:

**Case 1**

- Workload class 'MY_WORKLOAD_CLASS1' fit by mapping: PRIORITY 5, THREAD 5, MEMORY 50GB
• Workload class 'MY_WORKLOAD_CLASS2' fit by HINT: PRIORITY 4, THREAD 4, MEMORY 30GB
The hint clause is applied and the effective values are: PRIORITY 4, THREAD 4, and MEMORY 30 GB.

Case 2
• Workload class 'MY_WORKLOAD_CLASS1' fit by mapping: PRIORITY 5, THREAD 5, MEMORY 50GB
• Workload class 'MY_WORKLOAD_CLASS2' fit by HINT: PRIORITY 4, THREAD undefined, MEMORY 30GB
The hint clause is ignored because the thread limit is not defined.

Case 3
• Workload class 'MY_WORKLOAD_CLASS1' fit by mapping: PRIORITY 5, THREAD undefined, MEMORY 50GB
• Workload class 'MY_WORKLOAD_CLASS2' fit by HINT: PRIORITY 4, THREAD undefined, MEMORY 30GB
The hint clause is applied because both thread limits are not defined.

Case 4
• Workload class 'MY_WORKLOAD_CLASS1' fit by mapping: PRIORITY 5, THREAD undefined, MEMORY 50GB
• Workload class 'MY_WORKLOAD_CLASS2' fit by HINT: PRIORITY 4, THREAD undefined, MEMORY 70GB
The hint clause is ignored because the memory limit exceeded the 50 GB that is defined by the mapping.

Hints for Size Estimation

Size Estimations are related to the SQL Optimizer and are necessary to find an optimal plan.

ESTIMATION_SAMPLES(number)
• Controls the number of samples that are randomly picked from each of the base tables. The samples estimate local filter selectivity, as well as join condition selectivity. You can increase the number of samples to increase the accuracy of the estimation in exchange for a longer compilation time.
• By default, the <number> value is 1000 when there is no hint.
• If the <number> is 0, then no sampling is done and all sampling related estimation is skipped.
• A <number> value, which is below 0 is not allowed and throws an exception.

JOIN_SAMPLING, NO_JOIN_SAMPLING
• Turns join selectivity sampling on or off.
• By default join sampling is enabled.
• The feature also turns off when ESTIMATION_SAMPLES(0) is specified because there are no samples to use.

JOIN_SKEW_ESTIMATION, NO_JOIN_SKEW_ESTIMATION
• Turns join skew estimation using the top k value from base tables on or off.
• By default join skew estimation is enabled, but depending on the INI configuration, it can be turned off. The hint has higher priority over the INI configuration settings.
• Join skew estimation is internally implemented independently from join sampling. Join sampling uses randomly selected samples, whereas skew factor computations use the top 10 most frequent values.
provided by the column store. Their related issues have different characteristics: Join sampling related issues are usually performance related issues whereas Skew factor computation issues are usually crash problems.

**TABLE_FILTER_ESTIMATION(MINIMAL | SAMPLING | ALL)**

- Controls the level of techniques used to estimate the table filter selectivity. Lower-level use is a subset of techniques that are used in the upper level.
- MINIMAL: uses heuristic-based selectivity estimation using only base statistics (table row count and distinct count of each column).
- SAMPLING: uses basic sampling for filter estimation together with heuristic-based estimation.
- ALL (default): uses table filter estimation. Newly added estimation techniques are only included for this level.

**DISTRIBUTED_ESTIMATION, NO_DISTRIBUTED_ESTIMATION**

- Turns remote size statistics (row count/distinct count/filter selectivity of remote tables) retrieval on or off.
- By default remote statistics retrieval is enabled, but depending on the initial configuration, it can be turned off. The hint has higher priority over the initial configuration setting.

**EXHAUSTIVE_JOIN_ESTIMATION, NO_EXHAUSTIVE_JOIN_ESTIMATION**

Controls if the size estimation should be updated using join alternatives. While this increases the accuracy of the size estimation of join, it can also increase the compilation time and memory. The default value is off.

Examples:

```sql
SELECT * FROM T1 WITH HINT( ESTIMATION_SAMPLES(0) );
SELECT * FROM T1 WITH HINT( JOIN_SAMPLING );
SELECT * FROM T1 WITH HINT( NO_JOIN_SKEW_ESTIMATION );
SELECT * FROM T1 WITH HINT( TABLE_FILTER_ESTIMATION(SAMPLING) );
SELECT * FROM T1 WITH HINT( NO_DISTRIBUTED_ESTIMATION );
```

**Hints for SQL Optimizer Plan Generation**

There can be a plan where the same sequence of execution repeats multiple times, each with a different requester. For such plans, the optimizer can perform the sequence of executions only once and pass the results to all the requesters depending on the cost model (called subplan sharing).

**i Note**

Subplan sharing can be beneficial if the repeated sequence uses heavy CPU, consumes a large amount of memory, or takes a long time to execute.

**SUBPLAN_SHARING**

Guides the optimizer to prefer choosing the shared subplan.
• Accepts view names as optional parameters to provide more granular control of subplan sharing.
• Accepts aliases and nested views (optional).

**NO_SUBPLAN_SHARING**

Guides the optimizer to unfold the shared subplan:
• Accepts view names as optional parameters to provide more granular control of unfolding the shared subplan.
• Accepts aliases and nested views (optional).

Examples:

```sql
SELECT * FROM T1 WITH HINT( SUBPLAN_SHARING );
SELECT * FROM VIEW1 WITH HINT( SUBPLAN_SHARING (VIEW1) );
SELECT * FROM T1 WITH HINT( NO_SUBPLAN_SHARING );
SELECT * FROM VIEW1,VIEW2 WITH HINT( NO_SUBPLAN_SHARING (VIEW1, VIEW2) );
```

The following example forces subplan sharing on view1 and forces no subplan sharing on view2 (the rest of the shared views are generated following default logic):

```sql
SELECT * FROM VIEW1,VIEW2 WITH HINT( SUBPLAN_SHARING (VIEW1), NO_SUBPLAN_SHARING (VIEW2) );
```

The following example forces subplan sharing on view1 and forces no subplan sharing on any other possible shared views:

```sql
SELECT * FROM VIEWS WITH HINT( SUBPLAN_SHARING (VIEW1), NO_SUBPLAN_SHARING );
```

Calculation views are unfolded during plan generation to enable optimization using the SQL Optimizer whenever possible. Normally an unfolding calculation view is beneficial, but there can be cases where it leads to a degradation in performance.

**CALC_VIEW_UNFOLDING**

Guides the optimizer to unfold calculation views.
• Accepts view names as optional parameters to provide more granular control of calculation view unfolding.
• Accepts nested views but does not accept aliases (optional).

**NO_CALC_VIEW_UNFOLDING**

Guides the optimizer to preserve the calculation view (does not unfold the calculation view):
• Accepts view names as optional parameters to provide more granular control of calculation view unfolding.
• Accepts nested views but does not accept aliases (optional).

Examples:

```sql
SELECT * FROM T1 WITH HINT( CALC_VIEW_UNFOLDING );
SELECT * FROM VIEW1 WITH HINT( CALC_VIEW_UNFOLDING ("SYS_BIC"."CALC_VIEW") );
```
SELECT * FROM T1 WITH HINT (NO_CALC_VIEW_UNFOLDING);

SELECT * FROM VIEW1, VIEW2 WITH HINT (NO_CALC_VIEW_UNFOLDING (
"SYS_BIC"."CALC_VIEW1", "SYS_BIC"."CALC_VIEW2"));

The following example forces calculation view unfolding on view1 and forces no calculation view unfolding on view2 (the rest of the calculation views are generated following default logic):

SELECT * FROM VIEW1, VIEW2 WITH HINT (CALC_VIEW_UNFOLDING (
"SYS_BIC"."CALC_VIEW1"), NO_CALC_VIEW_UNFOLDING (
"SYS_BIC"."CALC_VIEW2"));

The following example forces calculation view unfolding on view1 and forces no calculation view unfolding on any other calculation views:

SELECT * FROM VIEWS WITH HINT (CALC_VIEW_UNFOLDING (
"SYS_BIC"."CALC_VIEW1"),
NO_CALC_VIEW_UNFOLDING);

There are optimization levels created during plan generation to enable optimization using the SQL Optimizer. Normally, a high optimization level is beneficial but selecting another level can be useful in some cases when optimization leads to a degradation in performance:

- Accepts ID-type arguments (MINIMAL | RULE_BASED | MINIMAL_COST_BASED | COST_BASED).
- COST_BASED is the default optimization level, which enables all optimization steps.
- Other hints that control rewriting rules can be used together with the OPTIMIZATION_LEVEL to selectively override decisions made by the hint.

### OPTIMIZATION_LEVEL

<table>
<thead>
<tr>
<th>Hint Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTIMIZATION_LEVEL (MINIMAL)</td>
<td>Disables all optimization rules except the mandatory rules that execute the query as it is.</td>
</tr>
<tr>
<td>OPTIMIZATION_LEVEL (RULE_BASED)</td>
<td>Displays all rewriting rules, including the MINIMAL optimization level.</td>
</tr>
<tr>
<td>OPTIMIZATION_LEVEL (MINIMAL_COST_BASED)</td>
<td>Displays the RULE_BASED optimization level plus the heuristic join reordering.</td>
</tr>
<tr>
<td>OPTIMIZATION_LEVEL (COST_BASED)</td>
<td>Displays the MINIMAL_COST_BASED optimization level plus available cost-based optimizations, including logical enumeration. This is the default option.</td>
</tr>
</tbody>
</table>

Examples:

- SELECT * FROM TABLES WITH HINT (OPTIMIZATION_LEVEL(MINIMAL))
- SELECT * FROM TABLES WITH HINT (OPTIMIZATION_LEVEL(MINIMAL), JOIN_REMOVAL)
- SELECT * FROM TABLES WITH HINT (OPTIMIZATION_LEVEL(RULE_BASED))
- SELECT * FROM TABLES WITH HINT (OPTIMIZATION_LEVEL(MINIMAL_COST_BASED))
Inter-operator parallelism for multiple column search operator is decided during plan generation to enable optimization using the SQL Optimizer. Normally parallelism is beneficial in performance, but minimizing parallelism is useful in some cases.

**PARALLEL_COLUMN_SEARCH**
Guides the optimizer to force column search operator parallelism.

**NO_PARALLEL_COLUMN_SEARCH**
Guides the optimizer to minimize column search operator parallelism.

Example:

```
SELECT * FROM (SELECT t1.a, t1.b, SUM(t1.c) AS s1, ROW_NUMBER() OVER (PARTITION by t1.b) AS rr1 FROM t1,t2 WHERE t1.a+0=t2.a+0 GROUP BY t1.a, t1.b) UNION ALL (SELECT t2.a, t2.b, SUM(t2.c) AS s2, ROW_NUMBER() OVER (PARTITION BY t2.b) AS rr2 FROM t2 GROUP BY t2.a, t2.b) WITH HINT (NO_CS_EXPR_JOIN, PARALLEL_COLUMN_SEARCH)
```

Generated plan:

Without the hint, TS1 is not parallelized, which means an OLTP search may be used. However, you can parallelize TS1 by using the hint PARALLEL_COLUMN_SEARCH.

The predicate is simplified during plan generation to enable optimization using the SQL Optimizer whenever possible. Normally, simplifying the predicate is beneficial but there can be cases where it leads to a degradation in performance.

**PRESERVE_GROUPBY**
Forces the SQL Optimizer to not remove or retrieve any redundant group bys in the plan and executes as many user-given group bys as possible. This aids in avoiding using up too much memory caused by large intermediate results made of joins.

This hint blocks the following transformations:
- Join through aggregation
- Merge aggregation
- Group by simplification

SELECT * FROM TABLES WITH HINT (OPTIMIZATION_LEVEL(COST_BASED), NO_JOIN_REMOVAL)
NO_PREDICATE_SIMPLIFICATION
Guides the optimizer to disable predicate simplification. This setting applies to all internal predicates.

Example:

| Given predicate: a != 0 and a = 0 Without hint: const false With hint: a != 0 and a = 0 |

Constant values are pre-evaluated during plan generation to enable optimization using the SQL Optimizer whenever possible. Normally pre-evaluating constant values are beneficial but there can be cases where it leads to a degradation in performance.

NO_CONST_PREEVALUATION
Guides the optimizer to disable constant pre-evaluation. This is applied to all internal constant values.

Example:

| Given constant value: a = 0 + 1 Without hint: a = 1 With hint: a = 0 + 1 |

JOIN_FILTER_REORDERING
Guides the optimizer to reorder join predicates inside of conjunctive and disjunctive conditions:

- Reordering is done by using the estimated selectivity of each predicate inside of conjunctive and disjunctive terms.
- Can only be applied to row store join operations.

OLAP_FACT_TABLE("table_name")
Chooses a fact table whose name is as given:

- If the <table_name> does not exist, then the hint is ignored.
- If candidate table exists more than two times in one column search (for example, in a self-join), then one of the tables is chosen by the current fact table decision heuristic logic.

(NO_)RECOMPILE_WITH_SQL_PARAMETERS
Guides the optimizer to enable/disable recompilation if a query has parameters:

- The default behavior for compilation is to compile without parameters.
- The default behavior for execution is to recompile with parameters if parameters exist and the plan is not based on heuristics.
- When used with the RECOMPILE_WITH_SQL_PARAMETERS hint, this is compiled without parameters.
- When used with the RECOMPILE_WITH_SQL_PARAMETERS hint, execution is performed by recompiling with parameters if parameters exist.
- When used with the NO_RECOMPILE_WITH_SQL_PARAMETERS hint, this is compiled without parameters.

NO_JOIN_CARDINALITY(option)
Discards the join cardinality information in the plan so that it does not affect the optimization.

<table>
<thead>
<tr>
<th>Hint name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_JOIN_CARDINALITY(ALL)</td>
<td>Discards all join cardinality information in the plan.</td>
</tr>
</tbody>
</table>
CHECK_JOIN_CARDINALITY

Forces the SQL Optimizer to create a plan that runs in join cardinality check mode. It checks every join operator with join cardinality information and returns an error when it finds any row that violates this information. The error message displays the identifier of the join operator with the violation. It can only find one join operator at a time so to correct all of the incorrect join cardinalities in a query it is necessary to fix one issue at a time and repeat the check until all of the errors are resolved.

**Note**

Due to the additional checking, the query performance can be very slow compared to the plan without the hint so it is not recommended to turn the checking on by default.

OPTIMIZATION_TRANSFORMATION_LIMIT(integer)

Sets the maximum number of transformation rules to apply during plan enumeration.

The default value is 2,000. Specifying a large value can lead to increased compilation time.

NO_LARGE_EXPR_MATERIALIZATION

Disables the early materialization of a large expression created by the SQL Optimizer.

The unfolding view copies the expression when the parent of the view references the view column. When the expression is very large, copying it can have a huge impact, slowing down other optimization steps because of the increase in size.

**Hints for Query Rewriting and Logical Transformations**

The SQL Optimizer relies heavily on query rewriting and logical transformation rules to enumerate all possible plans to find the best plan in the potential search space. Using the hints below can help you manipulate the SQL Optimizer and provide a quick workaround to fix various issues. For example:

```sql
SELECT * FROM T1 WITH HINT( JOIN_REMOVAL );
SELECT * FROM T1 WITH HINT( NO_JOIN_REMOVAL );
```

Refer to the below table for a list of hints and their descriptions:

<table>
<thead>
<tr>
<th>Category</th>
<th>Hint Name</th>
<th>Description</th>
<th>Available from</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>AGGR_TARGET</td>
<td>Prefers aggregation toward a target.</td>
<td>SAP HANA 2.0 SPS 04</td>
</tr>
<tr>
<td>Category</td>
<td>Hint Name</td>
<td>Description</td>
<td>Available from</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------</td>
<td>--------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>DISJ_FILTER_TRANSFORMATION</td>
<td>Prefers a split disjunctive filter into union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_DISJ_FILTER_TRANSFORMATION</td>
<td>Avoids a split disjunctive filter into union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>JOIN_SIMPLIFICATION</td>
<td>Prefers join simplification.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_JOIN_SIMPLIFICATION</td>
<td>Avoids join simplification.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>GROUPING_REMOVAL</td>
<td>Prefers unnecessary grouping removal.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_GROUPING_REMOVAL</td>
<td>Avoids unnecessary grouping removal.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>AGGR_SIMPLIFICATION</td>
<td>Prefers aggregation simplification.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_AGGR_SIMPLIFICATION</td>
<td>Avoids aggregation simplification.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>MATERIALIZED_COLUMN_REMOVAL</td>
<td>Prefers an unnecessary column removal.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_MATERIALIZED_COLUMN_REMOVAL</td>
<td>Avoids an unnecessary column removal.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>JOIN_REMOVAL</td>
<td>Prefers an unnecessary join removal.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_JOIN_REMOVAL</td>
<td>Avoids an unnecessary join removal.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>GROUPING_SIMPLIFICATION</td>
<td>Prefers a group by simplification.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_GROUPING_SIMPLIFICATION</td>
<td>Avoids a group by simplification.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>CONST_VIEW_UNFOLDING</td>
<td>Prefers constant view unfolding.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NOCONST_VIEW_UNFOLDING</td>
<td>Avoids constant view unfolding.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>HOST_PORT_JOIN_COLOCATION</td>
<td>Prefers a host/port join thru union.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_HOST_PORT_JOIN_COLOCATION</td>
<td>Avoids a host/port join thru union.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>CALC_VIEW_UNFOLDING</td>
<td>Prefers the unfold calculation view.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_CALC_VIEW_UNFOLDING</td>
<td>Avoids the unfold calculation view.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>TYPE_CAST_REMOVAL</td>
<td>Prefers the removal of a redundant type cast.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>Category</td>
<td>Hint Name</td>
<td>Description</td>
<td>Available from</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------</td>
<td>-------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_TYPE_CAST_REMOVAL</td>
<td>Avoids the removal of a redundant type cast.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>WINDOW_REMOVAL</td>
<td>Prefers the removal of window function aggregation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_WINDOW_REMOVAL</td>
<td>Avoids the removal of window function aggregation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>JOIN_REMOVAL_USING_CARDINALITY</td>
<td>Prefers join removal based on cardinality.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_JOIN_REMOVAL_USING_CARDINALITY</td>
<td>Avoids join removal based on cardinality.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>PARTITION_PRUNING</td>
<td>Prefers compile-time partition pruning.</td>
<td>SPS 11</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_PARTITION_PRUNING</td>
<td>Avoids compile-time partition pruning.</td>
<td>SPS 11</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>FILTER_RULE</td>
<td>Prefers applying the filter rule.</td>
<td>SPS 01</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_FILTER_RULE</td>
<td>Avoids applying the filter rule.</td>
<td>SPS 01</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>JOIN_RULE</td>
<td>Prefers applying the join rule.</td>
<td>SPS 01</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_JOIN_RULE</td>
<td>Avoids applying the join rule.</td>
<td>SPS 01</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>UNFOLD_SCALAR_UDF</td>
<td>Prefers scalar UDF unfolding.</td>
<td>SPS 04</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>NO_UNFOLD_SCALAR_UDF</td>
<td>Avoids scalar UDF unfolding.</td>
<td>SPS 04</td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td>SELF_JOIN_REMOVAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZER_REWRITE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENU-</td>
<td>AGGR_THRU_FILTER</td>
<td>Prefers push down aggregation through a filter.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>MERATION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENU-</td>
<td>NO_AGGR_THRU_FILTER</td>
<td>Avoids push down aggregation through a filter.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>MERATION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENU-</td>
<td>AGGR_THRU_JOIN</td>
<td>Prefers push down aggregation through join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>MERATION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENU-</td>
<td>NO_AGGR_THRU_JOIN</td>
<td>Avoids push down aggregation through join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>MERATION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENU-</td>
<td>DISTINCT_THRU_UNION</td>
<td>Prefers push down distinct through union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>MERATION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENU-</td>
<td>NO_DISTINCT_THRU_UNION</td>
<td>Avoids push down distinct through union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>MERATION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENU-</td>
<td>AGGR_INTO_AGGR</td>
<td>Prefers merging two aggregations into one.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>Category</td>
<td>Hint Name</td>
<td>Description</td>
<td>Available from</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------</td>
<td>--------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_AGGR_INTO_AGGR</td>
<td>Avoids merging two aggregations into one.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>AGGR_INTO_TABLE</td>
<td>Prefers merging aggregation with table scans.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_AGGR_INTO_TABLE</td>
<td>Avoids merging aggregation with table scans.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>DISTINCT_INTO_DISTINCT</td>
<td>Prefers merging two distincts into one.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_DISTINCT_INTO_DISTINCT</td>
<td>Avoids merging two distincts into one.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>DISTINCT_INTO_SEMI_JOIN</td>
<td>Prefers merging a distinct with a join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_DISTINCT_INTO_SEMI_JOIN</td>
<td>Avoids merging a distinct with a join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>UNION_INTO_UNION</td>
<td>Prefers merge_union_all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_UNION_INTO_UNION</td>
<td>Avoids merge_union_all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>UNION_INTO_UNION_THRU_AGGR</td>
<td>Prefers merge_union_all through aggregation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_UNION_INTO_UNION_THRU_AGGR</td>
<td>Avoids merge_union_all through aggregation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>PREAGGR_BEFORE_JOIN</td>
<td>Prefers pre-aggregation before join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_PREAGGR_BEFORE_JOIN</td>
<td>Avoids pre-aggregation before join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>DOUBLE_PREAGGR_BEFORE_JOIN</td>
<td>Prefers pre-aggregation before join for both of the children.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_DOUBLE_PREAGGR_BEFORE_JOIN</td>
<td>Avoids pre-aggregation before join for both of the children.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>PREAGGR_BEFORE_UNION</td>
<td>Prefers pre-aggregation before union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_PREAGGR_BEFORE_UNION</td>
<td>Avoids pre-aggregation before union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>DISJ_JOIN_INTO_UNION</td>
<td>Prefers to split disjunctive join predicates into filters and making union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_DISJ_JOIN_INTO_UNION</td>
<td>Avoids splitting disjunctive join predicates into filters and making union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>Category</td>
<td>Hint Name</td>
<td>Description</td>
<td>Available from</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>PREDISTINCT_BEFORE_SEMI_JOIN</td>
<td>Prefers putting DISTINCT on the right child of SEMI or ANTI SEMI joins.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_PREDISTINCT_BEFORE_SEMI_JOIN</td>
<td>Avoids putting DISTINCT on the right child of SEMI or ANTI SEMI joins.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>JOIN_THRU_AGGR</td>
<td>Prefers pushing down joins through aggregation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_JOIN_THRU_AGGR</td>
<td>Avoids pushing down joins through aggregation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>JOIN_THRU_FILTER</td>
<td>Prefers pushing down joins through the filter.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_JOIN_THRU_FILTER</td>
<td>Avoids pushing down joins through the filter.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>JOIN_THRU_JOIN</td>
<td>Prefers pushing down joins through join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_JOIN_THRU_JOIN</td>
<td>Avoids pushing down joins through join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>JOIN_THRU_UNION</td>
<td>Prefers pushing down joins through union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_JOIN_THRU_UNION</td>
<td>Avoids pushing down joins through union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>SEMI_JOIN_BEFORE_UNION</td>
<td>Prefers pushing down semi-joins through union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_SEMI_JOIN_BEFORE_UNION</td>
<td>Avoids pushing down semi-joins through union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>FILTER_THRU_JOIN</td>
<td>Prefers pushing down filters through join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_FILTER_THRU_JOIN</td>
<td>Avoids pushing down filters through join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>FILTER_THRU_AGGR</td>
<td>Prefers pushing down filters through aggregation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_FILTER_THRU_AGGR</td>
<td>Avoids pushing down filters through aggregation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>FILTER_THRU_UNION</td>
<td>Prefers pushing down filters through union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_FILTER_THRU_UNION</td>
<td>Avoids pushing down filters through union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>PRELIMIT_BEFORE_UNION</td>
<td>Prefers pushing down limits before union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_PRELIMIT_BEFORE_UNION</td>
<td>Avoids pushing down limits before union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>Category</td>
<td>Hint Name</td>
<td>Description</td>
<td>Available from</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>LIMIT_THRU_JOIN</td>
<td>Prefers pushing down limits through joins.</td>
<td>SPS 11</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_LIMIT_THRU_JOIN</td>
<td>Avoids pushing down limits through joins.</td>
<td>SPS 11</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>PRELIMIT_BEFORE_JOIN</td>
<td>Prefers LIMIT/SORT on the left child of the LEFT OUTER JOIN.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_PRELIMIT_BEFORE_JOIN</td>
<td>Avoids LIMIT/SORT on the left child of the LEFT OUTER JOIN.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>PREAMGR_BEFORE_CASE_AGGR</td>
<td>Prefers pre-aggregation before case expression aggregation.</td>
<td>SAP HANA 2.0 SPS 03</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_PREAMGR_BEFORE_CASE_AGGR</td>
<td>Avoids pre-aggregation before case expression aggregation.</td>
<td>SAP HANA 2.0 SPS 03</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>DOUBLE_JOIN_THRU_UNION_ALL</td>
<td>Prefers a pushdown join through double union all.</td>
<td>SAP HANA 2.0 SPS 03</td>
</tr>
<tr>
<td>OPTIMIZER_LOGICAL_ENUMERATION</td>
<td>NO_DOUBLE_JOIN_THRU_UNION_ALL</td>
<td>Avoids a pushdown join through double union all</td>
<td>SAP HANA 2.0 SPS 03</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>INDEX_SEARCH</td>
<td>Prefers the row engine index search.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_INDEX_SEARCH</td>
<td>Avoids the row engine index search.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>INDEX_JOIN</td>
<td>Prefers the row engine index join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_INDEX_JOIN</td>
<td>Avoids the row engine index join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>HASH_JOIN</td>
<td>Prefers the row engine hash join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_HASH_JOIN</td>
<td>Avoids the row engine hash join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>RANGE_JOIN</td>
<td>Prefers the row engine range join.</td>
<td>SPS 11</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_RANGE_JOIN</td>
<td>Avoids the row engine range join.</td>
<td>SPS 11</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>HASHED_RANGE_JOIN</td>
<td>Prefers the row engine hashed range join.</td>
<td>SPS 11</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_HASHED_RANGE_JOIN</td>
<td>Avoids the row engine hashed range join.</td>
<td>SPS 11</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>JOIN_FILTER_REORDERING</td>
<td>Prefers join predicate reordering.</td>
<td>SPS 11</td>
</tr>
<tr>
<td>Category</td>
<td>Hint Name</td>
<td>Description</td>
<td>Available from</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------</td>
<td>--------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_JOIN_FILTER_REORDERING</td>
<td>Avoids join predicate reordering.</td>
<td>SPS 11</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>MIXED_INVERTED_INDEX_JOIN</td>
<td>Prefers mixed index join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_MIXED_INVERTED_INDEX_JOIN</td>
<td>Avoids mixed index join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_SORT</td>
<td>Prefers the column engine order by.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_SORT</td>
<td>Avoids the column engine order by.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_LIMIT</td>
<td>Prefers the column engine limit.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_LIMIT</td>
<td>Avoids the column engine limit.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_FILTER</td>
<td>Prefers the column engine filter.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_FILTER</td>
<td>Avoids the column engine filter.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_DISTINCT</td>
<td>Prefers the column engine distinct.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_DISTINCT</td>
<td>Avoids the column engine distinct.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_AGGR</td>
<td>Prefers column engine aggregation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_AGGR</td>
<td>Avoids column engine aggregation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_JOIN</td>
<td>Prefers column engine join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_JOIN</td>
<td>Avoids the column engine join.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_SCALAR_SUBQUERY</td>
<td>Prefers the column engine scalar subquery.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_SCALAR_SUBQUERY</td>
<td>Avoids the column engine scalar subquery.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_ITAB_IN_SUBQUERY</td>
<td>Prefers the column engine ITAB in the subquery.</td>
<td>SPS 09 (extended to normal column tables in SPS 12)</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_ITAB_IN_SUBQUERY</td>
<td>Avoids the column engine ITAB in the subquery.</td>
<td>SPS 09 (extended to normal column tables in SPS 12)</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_ITAB_IN_PREDICATE</td>
<td>Prefers the column engine ITAB for multi-columns in predicates.</td>
<td>SPS 12</td>
</tr>
<tr>
<td>Category</td>
<td>Hint Name</td>
<td>Description</td>
<td>Available from</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_ITAB_IN_PREDICATE</td>
<td>Avoids the column engine ITAB for multi-columns in predicates.</td>
<td>SPS 12</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_UNION_ALL</td>
<td>Prefers the column engine union all.</td>
<td>SPS 09 (extended to utilize new column union all in SP12)</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_UNION_ALL(CALC)</td>
<td>Prefers the calc engine union all.</td>
<td>SPS 12 (extended to utilize the old CE column union all in SP12)</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_UNION_ALL</td>
<td>Avoids the column engine union all.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_WINDOW</td>
<td>Prefers the column engine window for row numbers only.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_WINDOW</td>
<td>Avoids the column engine window for row numbers only.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CYCLIC_JOIN</td>
<td>Prefers cyclic joins in a single column search.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CYCLIC_JOIN</td>
<td>Avoids cyclic joins in a single column search.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>OLAP_GUIDED_NAVIGATION</td>
<td>Prefers OLAP engine guided navigation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_OLAP_GUIDED_NAVIGATION</td>
<td>Avoids OLAP engine guided navigation.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>OLAP_FEMS</td>
<td>Prefers the OLAP engine fems.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_OLAP_FEMS</td>
<td>Avoids the OLAP engine fems.</td>
<td>SPS 09</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_EXPR_JOIN</td>
<td>Prefers the column engine expression join.</td>
<td>SPS 10</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_EXPR_JOIN</td>
<td>Avoids the column engine expression join.</td>
<td>SPS 10</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_SEMI_JOIN</td>
<td>Prefers the column engine semi-join.</td>
<td>SPS 12</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_SEMI_JOIN</td>
<td>Avoids the column engine semi-join.</td>
<td>SPS 12</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>CS_OUTER_JOIN_WITH_FILTER</td>
<td>Prefers the outer join with the filter pushed down to the column engine.</td>
<td>SPS 12</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_CS_OUTER_JOIN_WITH_FILTER</td>
<td>Avoids the outer join with the filter pushed down to the column engine.</td>
<td>SPS 12</td>
</tr>
<tr>
<td>Category</td>
<td>Hint Name</td>
<td>Description</td>
<td>Available from</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>REMOTE_COLUMN_SCAN</td>
<td>Prefers the ITAB-producing remote scan operator.</td>
<td>SPS 10</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_REMOTE_COLUMN_SCAN</td>
<td>Avoids the ITAB-producing remote scan operator.</td>
<td>SPS 10</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>REMOTE_JOIN_RELOCATION</td>
<td>Prefers join relocation. Performs a JOIN between SAP HANA and the remote table at the remote database.</td>
<td>SPS 10</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_REMOTE_JOIN_RELOCATION</td>
<td>Avoids join relocation. Performs a JOIN between SAP HANA and the remote table at the remote database.</td>
<td>SPS 10</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>REMOTE_EXPR_MATERIALIZATION</td>
<td>Prefers expression materialization at the remote database.</td>
<td>SPS 10</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_REMOTE_EXPR_MATERIALIZATION</td>
<td>Avoids expression materialization at the remote database.</td>
<td>SPS 10</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>DISJUNCTIVE_HASH_JOIN</td>
<td>Prefers disjunctive hash join implementation.</td>
<td>SAP HANA 2.0 SPS 00</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_DISJUNCTIVE_HASH_JOIN</td>
<td>Avoids disjunctive hash join implementation.</td>
<td>SAP HANA 2.0 SPS 00</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>OLAP_FACT_TABLE</td>
<td>Prefers choosing a fact table whose name is the same as is given.</td>
<td>SPS 12</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_TOPK_SORT</td>
<td>Prefers choosing a fact table whose name is the same as the one given.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_ESX_TOPK_SORT</td>
<td>Avoids choosing a fact table whose name is the same as the one given.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_FILTER</td>
<td>Prefers using the ESX engine filter.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_ESX_FILTER</td>
<td>Avoids using the ESX engine filter.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_SORT</td>
<td>Prefers using the ESX engine sort.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_ESX_SORT</td>
<td>Avoids using the ESX engine sort.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_LIMIT</td>
<td>Prefers using the ESX engine limit.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_ESX_LIMIT</td>
<td>Avoids using the ESX engine limit.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>Category</td>
<td>Hint Name</td>
<td>Description</td>
<td>Available from</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------</td>
<td>--------------------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_DISTINCT</td>
<td>Prefers using the ESX engine distinct.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_ESX_DISTINCT</td>
<td>Avoids using the ESX engine distinct.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_AGGR</td>
<td>Prefers using ESX engine aggregation.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_ESX_AGGR</td>
<td>Avoids using ESX engine aggregation.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_JOIN</td>
<td>Prefers using the ESX join.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_ESX_JOIN</td>
<td>Avoids using the ESX join.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_UNION_ALL</td>
<td>Prefers using the ESX engine union all.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_ESX_UNION_ALL</td>
<td>Avoids using the ESX engine union all.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_WINDOW</td>
<td>Prefers using the ESX engine window.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_ESX_WINDOW</td>
<td>Avoids using the ESX engine window.</td>
<td>SAP HANA 2.0 SPS 02</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_DISK_HASH_JOIN</td>
<td>Prefers ESX engine disk-based hash join.</td>
<td>SAP HANA 2.0 SPS 04</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>NO_ESX_DISK_HASH_JOIN</td>
<td>Avoids ESX engine disk-based hash join.</td>
<td>SAP HANA 2.0 SPS 04</td>
</tr>
<tr>
<td>OPTIMIZER_PHYSICAL_ENUMERATION</td>
<td>ESX_PARTITION_HASH_JOIN_BUCKETS(&lt;bucket_size&gt;)</td>
<td>Sets the size of the bucket for an ESX disk-based hash join. The hint configures the number of buckets to split the data.</td>
<td>SAP HANA 2.0 SPS 04</td>
</tr>
</tbody>
</table>

**Related Information**

- HINTS System View [page 1585]
- SELECT Statement (Data Manipulation) [page 1104]
- STATEMENT_HINTS System View [page 1661]
- Using Hints to Query Data Snapshots
4.10.1.153 SET HISTORY SESSION Statement (Session Management)

Allows the current database session to view a previous version of history tables.

Syntax

```
SET HISTORY SESSION TO <when>
```

Elements

when

Specifies the time period of the history table that should be used in the current database session.

```
<when> ::=  
  NOW  
  COMMIT ID <commit_id>  
  UTCTIMESTAMP <timestamp>
```

NOW

Specifies that the database session uses the current version of the history table.

COMMIT ID commit_id

Specifies that the database session uses a version of the database table that was available at a specific commit ID.

```
COMMIT ID <commit_id>  
<commit_id> ::= <unsigned_integer>
```

UTCTIMESTAMP timestamp

Specifies that the database session uses a version of the database table that was available at a specific timestamp.

```
<timestamp> ::= <string_literal>
```

The timestamp must be in the format 'YYYY-MM-DD HH:MM:SS[.FF7]'.

Description

Specify the required table version by COMMIT ID or UTCTIMESTAMP format. You can also set the session to see the current state of history tables by using the NOW parameter.

This command only affects history tables; normal tables remain unchanged. This command cannot be executed in autocommit on mode or no-log-flush mode.
As an alternative to using SET HISTORY SESSION, you can also use statement-level time travel.

**Examples**

Create a history column on table x so you can view the table’s history.

```sql
CREATE HISTORY COLUMN TABLE x ( a int, b int ); // after turning off auto commit
```

Insert values into table x.

```sql
INSERT INTO x VALUES (1,1); COMMIT;
INSERT INTO x VALUES (2,2); COMMIT;
```

Obtain the commit ID of the last commit.

```sql
SELECT last_commit_id
FROM M_HISTORY_INDEX_LAST_COMMIT_ID
WHERE session_id = current_connection;
```

Before deleting the data, view the history of table x and perform a select.

View the history of table x using a timestamp and perform a select. Replace `<last_commit_id>` with the value you obtained from the previous query.

```sql
SET HISTORY SESSION TO COMMIT ID `<last_commit_id>`;
SELECT * FROM x;
```

Delete all the data from table x.

```sql
SET HISTORY SESSION TO NOW;
DELETE FROM x;
COMMIT;
```

The select query above returns the two records (1,1) and (2,2), which existed before you deleted the contents of table x.

Obtain the `<last_commit_id>` timestamp.

```sql
SELECT COMMIT_TIME FROM SYS.TRANSACTION_HISTORY WHERE commit_id = `<last_commit_id>`;
```

View the history of table x using a timestamp and perform a select. Replace `<commit_time>` with the value you obtained from the previous query.

```sql
SET HISTORY SESSION TO UTCTIMESTAMP '<commit_time>';
SELECT * FROM x;
```

The select query above returns the two records (1,1) and (2,2), which existed before you deleted the contents of table x.

Set the history session to the current commit state and perform a select.

```sql
SET HISTORY SESSION TO NOW;
```
The select query returns an empty table because, at the current state, the table has had all of its data deleted.

Related Information

- Data Types [page 33]
- M_HISTORY_INDEX_LAST_COMMIT_ID System View [page 1927]
- TRANSACTION_HISTORY System View [page 1693]
- HISTORY COLUMN Option (Time Travel) [page 875]

4.10.1.154 SET PSE Statement (System Management)

Sets a purpose for a PSE store.

Syntax

```sql
SET PSE <pse_name> PURPOSE <purpose> [ FOR <purpose_object_list> ]
```

Syntax Elements

- **pse_name**
  
  Specifies the name of the PSE store.

- **PURPOSE purpose**
  
  Specifies the purpose of the PSE.

```sql
<purpose> ::= JWT | SAML | SAP LOGON | SSL | X509 | LDAP | SOLACE | DATABASE_REPLICATION | REMOTE_SOURCE
```

- **purpose_object_list**
Specifies the providers, remote sources, and hosts to assign to an existing PSE that has no purpose objects assigned.

```sql
<purpose_object_list> ::= { PROVIDER <provider_list> | HOST <host_list> | REMOTE SOURCE <remote_source_list> }
<provider_list> ::= <provider> [ , <provider>[ ,... ] ]
<provider> ::= <simple_identifier>
:host_list> ::= <host> [ ,<host>[ ,... ] ]
<remote_source_list> ::= <remote_source> [ ,<remote_source> [,... ] ]
<remote_source> ::= <simple_identifier>
```

**Permissions**

If you are not the owner of the PSE store, then you must have the REFERENCES privilege on the PSE object. For the SSL purpose, you must also have the SSL ADMIN system privilege. For the SAML or JWT purposes, you must also have the USER ADMIN system privilege. For all other authentication purposes, you must also have the TRUST ADMIN privilege. To set the REMOTE SOURCE purpose you must have the system privilege CREATE SOURCE.

Duplicate definitions in `<provider_list>`, `<remote_source>`, or `<host_list>` are ignored.

Host names are case insensitive and should exclude the port number (for example, example.sap.corp). The host name format is validated, but the existence in the topology is not. If duplicate host names are present, then the first match from the SNI context list is used. The same host name can be assigned to multiple PSEs.

Multiple providers, remote sources, or hosts can be assigned to a single purpose, but a provider, remote source, or host can only be assigned to one PSE.

When assigning a provider, remote source, or host to a PSE, if the purpose provider or host already exists for another PSE, then the provider, remote source, or host is reassigned to the new PSE. If this results in no remaining purpose providers, remote source, or hosts for the original PSE, then the PSE purpose is removed from the original PSE.

For an existing PSE, to remove all assigned providers, remote source, or hosts from a purpose, but keep the purpose, execute the SET PSE statement without the FOR `<purpose_object_list>` clause. The purpose of the PSE becomes unqualified.

To remove specific providers, remote source, or hosts from a PSE, use ALTER PSE with the `<drop_purpose_object_clause>` clause.

For JWT and SAML, the provider must exist before it can be added. See `CREATE JWT PROVIDER Statement` and `CREATE SAML PROVIDER Statement`. For REMOTE SOURCE, the remote source must exist before it can be added. See `CREATE REMOTE SOURCE Statement`.

If you are not the owner of the PSE store, then you must have the REFERENCES privilege on the PSE object. For the SSL purpose, you must also have the SSL ADMIN system privilege. For the SAML or JWT purposes, you must also have the USER ADMIN system privilege. For all other authentication purposes, you must also have the TRUST ADMIN privilege. To set the REMOTE SOURCE purpose you must have the system privilege CREATE SOURCE.
Description

The purpose of a PSE can be qualified or unqualified. A PSE with an unqualified purpose has no provider or host assigned. This kind of PSE is used for authentication when no other PSE with the given purpose matches the provider or host of an incoming authentication request. A PSE with a qualified purpose has one or more providers or hosts assigned. This kind of PSE is preferred over a PSE with an unqualified purpose.

A PSE store with purpose JWT must contain a valid X509 certificate to validate the JWT.

To change the purpose of an existing PSE, you must first execute the UNSET PSE statement to remove the existing purpose from the PSE and then use the SET PSE statement to add a new purpose for the PSE.

Examples

Set the purpose SAP LOGON for the example_pse PSE store:

```
SET PSE example_pse PURPOSE SAP LOGON;
```

Set the SSL purpose for the PSE1 PSE store using host host1.sap.corp

```
SET PSE PSE1 PURPOSE SSL FOR HOST 'host1.sap.corp';
```

Set the REMOTE SOURCE purpose for the HDD_PSE store using HDB_SYS.

```
SET PSE HDD_PSE PURPOSE REMOTE SOURCE FOR REMOTE SOURCE HDB_SYS;
```

Related Information

- CREATE JWT PROVIDER Statement (Access Control) [page 767]
- CREATE SAML PROVIDER Statement (Access Control) [page 803]
- ALTER PSE Statement (System Management) [page 468]
- PSE_PURPOSE_OBJECTS System View [page 1621]
- CREATE REMOTE SOURCE Statement (Access Control) [page 799]
- SET PSE Statement (System Management) [page 1167]
- UNSET [SESSION] Statement (Session Management) [page 1183]
4.10.1.155 SET SCHEMA Statement (Session Management)

Changes the default schema for the session to the specified schema.

Syntax

```
SET SCHEMA <schema_name>
```

Syntax Elements

```
schema_name
```

Specifies the name of the schema that the session should change to.

```
<schema_name> ::= <unicode_name>
```

If `<schema_name>` is the name of a schema synonym, then the current schema is set to the schema that is referenced by the schema synonym.

Description

The user’s schema is used as the default schema when executing a SQL statement that references database objects without specifying a schema name.

Example

The following example creates a new schema called MY_SCHEMA, and then sets the default schema to it.

```
CREATE SCHEMA MY_SCHEMA;
SET SCHEMA MY_SCHEMA;
```

Now, if a table is created (for example, MY_TABLE) and no schema is specified, it is automatically created in the MY_SCHEMA schema. Likewise, a query such as `SELECT * FROM MY_TABLE;` would be equivalent to `SELECT * FROM MY_SCHEMA.MY_TABLE`. 
4.10.1.156  SET [SESSION] Statement (Session Management)

Sets a session variable for the current session.

Syntax

```
SET [SESSION] <variable_string_literal> = <value_string_literal>
```

Syntax Elements

- **variable**
  Specifies the variable to set. The maximum length of `<variable>` is 5000 characters.
- **value**
  Specifies the value for the specified variable. The maximum length of `<value>` is 5000 characters.

Description

The maximum number of session variables is 1024 per session by default (or as defined by the `max_session_variables` setting in the `indexserver.ini` configuration file.

Session variable settings may be overridden by options specified in a query.

Session variables with the `xs_` prefix are protected and cannot be overwritten with the standard SQL SET command.

Session variables can be retrieved by using the `SESSION_CONTEXT` function or by querying the `M_SESSION_CONTEXT` system view.

Example

Set the session variable MY_VAR to abc:

```
SET 'MY_VAR' = 'abc';
```

Select the variable MY_VAR from the current session.

```
SELECT SESSION_CONTEXT('MY_VAR') FROM DUMMY;
```
Related Information

M_SESSION_CONTEXT System View [page 2133]
UNSET [SESSION] Statement (Session Management) [page 1183]
SESSION_CONTEXT Function (Miscellaneous) [page 341]
Session Variables [page 75]

4.10.1.157 SET SYSTEM LICENSE Statement (System Management)

Installs a license key to the database instance.

Syntax

SET SYSTEM LICENSE <license_key>

Syntax Elements

license_key
Specifies the license key.

<string_literal>

Permissions

The LICENSE ADMIN system privilege is required to execute this command.
Description

The license key (license key="" ) is copied and pasted from the license key file.

Example

Set the system license with a license code. The license shown below does not contain a license key, so you cannot run this example.

```sql
SET SYSTEM LICENSE
  '----- Begin SAP License -----'
  SAPSYSTEM=HD1
  HARDWARE-KEY=K4150485960
  INSTNO=0110008649
  BEGIN=20110809
  EXPIRATION=20151231
  LKEY=....
  SWPRODUCTNAME=SAP-HANA
  SWPRODUCTLIMIT=2147483647
  SYSTEM-NR=00000000031047460';
```

4.10.1.158   SET TRANSACTION Statement (Transaction Management)

Sets transaction parameters.

Syntax

```sql
SET TRANSACTION 
  <isolation_level> 
  <transaction_access_mode> 
  <transaction_lock_wait_timeout> 
  <ddl_on_off>
```

Syntax Elements

**isolation_level**

Sets the statement level read consistency of the data in the database.

```sql
<isolation_level> ::= ISOLATION LEVEL <level>
<level> ::= 
  READ COMMITTED
```
If `<isolation_level>` is omitted, then the default is READ COMMITTED.

The READ COMMITTED isolation level provides statement-level read consistency during a transaction. Each statement in a transaction uses the committed state of the data in the database as the execution of the statement begins. This means that each statement in the same transaction may see varying snapshots of the data in the database as they are executed. This is because data can be committed during the transaction.

The REPEATABLE READ and SERIALIZABLE isolation levels provide transaction level snapshot isolation; that is, all statements of a transaction use the same snapshot of the database data. This snapshot contains all changes that were committed at the time the transaction started along with the changes made by the transaction itself.

REPEATABLE READ guarantees that the data the transaction reads is not being modified by another transaction and will not change unless the reading transaction modifies it and commits the change.

SERIALIZABLE guarantees maximum data integrity, since only one transaction run as a single serial operation can both read and modify the data at a time. Other transactions can access the data only after the serializable transaction has committed or rolled back. SERIALIZABLE can impact overall performance, since transactions are serialized.

### transaction_access_mode

Controls whether a transaction can modify data during execution. If transaction_access_mode is not set the default is READ WRITE.

```plaintext
<transaction_access_mode> ::= READ ONLY | READ WRITE
```

When READ ONLY access mode is set, only read operations with SELECT statements are allowed. This setting remains in effect for the session, even after a COMMIT or ROLLBACK. An exception is thrown if update or insert operations are attempted while in this mode.

When READ WRITE access mode is set, statements within a transaction can freely read or make changes to the database data as required.

### transaction_lock_wait_timeout

Configures a time limit for how long TABLE LOCK or RECORD LOCK waits.

```plaintext
<transaction_lock_wait_timeout> ::= LOCK WAIT TIMEOUT <unsigned_integer>
```

The default value of `<transaction_lock_wait_timeout>` is specified in `indexserver.ini`, in the transaction section. The value must specify the number of milliseconds and be between 0 and 2147483647. If a lock is not acquired before the timeout period expires, then the executing statement returns the error code 131. The transaction is rolled back by `<transaction_lock_wait_timeout>`.

### ddl_on_off

Turns transaction DDL on or off for the session.

```plaintext
<ddl_on_off> ::= DDL { ON | OFF }
```

When DDL OFF is specified, the session blocks all DDL executions. Also, the TRUNCATE statement works for local temporary tables but not with any other table type. The default behavior is DDL ON.
DCL statements like GRANT are threaded as DDL. Therefore, DCL statements are blocked when DDL blocking is enabled.

Query the value of M_SESSION_CONTEXT.DDL_ENABLED to determine the current setting for the session.

**Description**

The SAP HANA database uses multi-version concurrency control (MVCC) to ensure consistent read operations. Concurrent read operations have a consistent view of the database data without blocking concurrent write operations. Updates are implemented by inserting new versions of data and not by overwriting existing records.

The isolation level specified determines the lock operation type that is used. The system supports both statement level snapshot isolation and transaction level snapshot isolation.

For statement snapshot isolation use level READ COMMITTED. For transaction snapshot isolation use REPEATABLE READ or SERIALIZABLE.

During a transaction when rows are inserted, updated, or deleted, the system sets exclusive locks on the affected rows for the duration of the transaction. The system also sets shared locks on the affected tables for the duration of the transaction. This guarantees that the table is not dropped or altered while rows of the table are being updated. The database releases these locks at the end of the transaction.

Reading a row does not set any locks on either tables or rows within the database regardless of the isolation level used.

Data Definition Language (DDL) statements (CREATE TABLE, DROP TABLE, CREATE VIEW, and so on) always take an instantaneous effect on following SQL statements regardless of the transaction isolation level being used. For an example of this behavior, consider the following sequence:

1. A long running SERIALIZABLE isolation transaction begins operating on Table C.
2. From outside the transaction some DDL is executed which adds a new column to Table C.
3. From within the SERIALIZABLE isolation transaction the new column is accessible as soon as the DDL statement completes. This access occurs regardless of the isolation level of the transaction.

**Example**

The following example sets the transaction isolation level to READ COMMITTED to provide statement level read consistency during the current transaction.

```sql
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

The following example sets the transaction lock timeout to 1 minute. (1 second = 1000 milliseconds).

```sql
SET TRANSACTION LOCK WAIT TIMEOUT 60000;
```

The following example turns DDL off for the session and then turns it back on again:

```sql
CREATE TABLE Table0 (A INT);
SET TRANSACTION DDL OFF; --> Turns off DDL for the session
```
CREATE TABLE Table1 (A INT); --> DDL operation fails and returns an error indicating that DDL is disabled in the session
INSERT INTO Table0 VALUES (0); --> DML operation succeeds
SELECT VALUE FROM M_SESSION_CONTEXT WHERE KEY='DDL_ENABLED'; --> This returns FALSE for DDL_ENABLED.
SET TRANSACTION DDL ON; --> Turns DDL back on for the session

4.10.1.159 SET TRANSACTION AUTOCOMMIT DDL Statement (Transaction Management)

Specifies the auto commit property for DDL statements specific to the session.

Syntax

SET TRANSACTION AUTOCOMMIT DDL { ON | OFF }

Description

The default is ON. A user can specify the auto commit property for DDL statements specific to the session. If the auto commit property is on, then the transaction commits implicitly after executing each DDL statement. Otherwise, the transaction commits after executing a commit statement, and rollbacks of DDL statements are supported.

Execution of this command executes an implicit commit when the auto commit property is changed. This supports the changing of this property in a single transaction.

Partial rollback of DML is possible during an AUTOCOMMIT DDL OFF session, provided that no DDL was executed as part of the transaction.

Partial rollback of DDL is not possible during an AUTOCOMMIT DDL OFF session; any failure during an AUTOCOMMIT DDL OFF session that contains DDL, including uniqueness violations, triggers a full rollback.

For example:

SET TRANSACTION AUTOCOMMIT DDL OFF;
CREATE TABLE T1 (A INT PRIMARY KEY); <- Successful DDL
INSERT INTO T1 VALUES (1); <- Successful DML
INSERT INTO T1 VALUES (1); <- Uniqueness violation, triggering a full rollback due to the DDL in the transaction
SELECT * FROM T1; <- returns an invalid table name error due to the full rollback

There are some DDL cases that do not affect the rollback behavior, such as DDL that fails at syntax check, or DDL that fails due to a reference to a non-existent object. These cases are considered as ‘DDL is not executed’, and would not qualify the transaction for full rollback in the event of some other failure in the transaction.

Supportability of DDL rollbacks for various table types is as follows:

- Column, history column and row table: Rollbacks on DDL and DML statements are supported.
- Global temporary and global temporary column table: Rollbacks on DDL are supported, but rollbacks on DML for global temporary column tables are not supported.
- Local temporary and local temporary column table: Rollbacks on DDL are supported, but rollbacks on DML for local temporary column tables are not supported.
- No logging column table: Rollbacks on DDL are not supported, but rollbacks on DML are supported.

Rollbacks on the following DDL are not supported:
- ALTER TABLE {SET | UNSET} ROW ORDER
- ALTER TABLE to enable, disable client-side encryption or to change column encryption keys
- DDL on extended storage and extended storage tables
- {CREATE | ALTER | DROP} ADAPTER
- {CREATE | ALTER | DROP} AGENT

Before attempting to execute SET TRANSACTION AUTOCOMMIT DDL OFF, ensure that you are using an autocommit-disabled session.

**Example**

Turn off DDL autocommit mode.

```sql
SET TRANSACTION AUTOCOMMIT DDL OFF;
```

Create a table T and insert two rows into it.

```sql
CREATE ROW TABLE T (KEY INT PRIMARY KEY, VAL INT);
INSERT INTO T VALUES (1, 1);
INSERT INTO T VALUES (2, 2);
```

Commit the current transaction.

```sql
COMMIT;
```

Add a new column VAL2 to table T and insert a row.

```sql
ALTER TABLE T ADD (VAL2 INT);
INSERT INTO T VALUES (3, 3, 3);
```

Roll back the current transaction.

```sql
ROLLBACK;
```

Select the data in table T and get two rows with two columns.

```sql
SELECT * FROM T;
```
4.10.1.160  TRUNCATE TABLE Statement (Data Manipulation)

Deletes all rows from a table or projection view.

Syntax

```sql
TRUNCATE TABLE <table_name> [ PARTITION (<partition_list>) ]
```

Syntax Elements

- **table_name**
  Specifies the name of a table or projection view.

  ```sql
  <table_name> ::= [[<database_name>.]<schema_name>.]<identifier>
  <schema_name> ::= <unicode_name>
  ```

  For linked database, `<database_name>` is the name of the remote source. `<identifier>` is the name of the table on the remote source.

- **partition_list**
  Specifies the list of logical partition IDs of the partitions to truncate.

  ```sql
  <partition_list> ::= <partition_id> [, <partition_list>]  
  <partition_id> ::= <unsigned_integer>
  ```

  This can be done only for column store tables. Partitions in remote tables (those from a different tenant) and replicated tables (with either an OSTR/ATR replica) cannot be truncated.

  Logical partition IDs are listed in the PART_ID column in TABLE_PARTITIONS.

Description

TRUNCATE TABLE is faster than DELETE FROM when deleting all records from a table, but TRUNCATE TABLE cannot be rolled back. To roll back from record deletion, use the DELETE statement.

HISTORY tables can also be truncated by using this statement. All parts of the history table (main, delta, history main, and history delta) are truncated and the content is permanently deleted.

TRUNCATE TABLE causes an implicit commit.

TRUNCATE TABLE is also supported in DDL auto commit mode OFF, except for local and global temporary column tables. A TRUNCATE TABLE on temporary column tables cannot be rolled back, but it does not commit changes on other database objects.

In the documentation, TRUNCATE TABLE is included with the DML statements because its purpose is to delete table data and users typically associate it with a DML statement. However, TRUNCATE TABLE behaves as a
DDL statement (that is, it causes an implicit commit) if run when the transaction mode AUTOCOMMIT DDL is set to ON (SET TRANSACTION statement).

SQLScript transactions normally run in AUTOCOMMIT DDL OFF, so an implicit commit is not performed for TRUNCATE TABLE.

TRUNCATE TABLE is allowed on a history table associated with a system-versioned table. When you execute a TRUNCATE TABLE statement on the system-versioned table, truncated records are not archived in the associated history table.

Example

Create table A.

```sql
CREATE TABLE A (A INT PRIMARY KEY, B INT);
```

Truncate the contents of table A.

```sql
TRUNCATE TABLE A;
```

Create table T1.

```sql
CREATE COLUMN TABLE T1 (A INT, B INT) PARTITION BY RANGE (A) ((PARTITION 1 <= VALUES < 5, PARTITION 5 <= VALUES < 20, PARTITION OTHERS) SUBPARTITION BY RANGE (B) (PARTITION 1 <= VALUES < 100, PARTITION 100 <= VALUES < 200));
```

Truncate logical partitions 2 and 3.

```sql
TRUNCATE TABLE T1 PARTITION (2,3);
```

Related Information

DELETE Statement (Data Manipulation) [page 938]

4.10.1.161 UNLOAD Statement (Data Manipulation)

Unloads the column store table from memory.

Syntax

```sql
UNLOAD <table_name> [ { <partition_restriction> | <persistent_memory_clause> } ]
```
Syntax Elements

table_name

Unloads the specified table from memory.

```
<table_name> ::= [<schema_name>.[]<identifier>
```

partition_restriction

Unloads the specified partition(s) from memory. This option is supported for partitioned column store tables. For multistore tables, only partitions in default storage are supported. Partitions in extended storage are not supported and return an error if specified.

```
<partition_restriction> ::=   PARTITION ( <partition_number> [ <persistent_memory_clause> ] [,,
<partition_number> [ <persistent_memory_clause> ] [, ...] ] )
```

<partition_number> is the logical partition ID and is stored in the M_CS_PARTITIONS system view.

If <partition_restriction> is not specified, then all partitions are unloaded by default.

persistent_memory_clause Specifies whether persistent memory is deleted or retained when unloading a table.

```
<persistent_memory_clause> ::= { DELETE | RETAIN } PERSISTENT MEMORY
```

The default behavior is to retain the data files for real persistent memory and to delete the data files when persistent memory is configured on tmpfs.

- **DELETE** Deletes persistent memory for the column as part unloading the table.
- **RETAIN** Retains persistent memory for the column as part of unload table.

Description

To free up memory, use the UNLOAD statement to unload the column store table from memory. The table is loaded again on next access.

While the UNLOAD statement allows you to specify partitioned tables, the LOAD statement does not.

For more information on table partitioning, see the *SAP HANA Administration Guide*.

Permissions

You must have either the UPDATE privilege or the system privilege TABLE ADMIN to execute the UNLOAD statement on a table.
**Example**

Create column table A1.

```sql
CREATE COLUMN TABLE A1 (A INT, B INT);
```

Load table A1 into memory.

```sql
LOAD A1 all;
```

Unload table A1 from memory.

```sql
UNLOAD A1;
```

Check the load status of table A1.

```sql
SELECT loaded FROM m_cs_tables WHERE table_name = 'A1';
```

Delete persistent memory on partitions 1, but retain persistent memory on partition 2 when unloading table A1.

```sql
UNLOAD A1 PARTITION (1 DELETE PERSISTENT MEMORY, 2 RETAIN PERSISTENT MEMORY);
```

Delete all persistent memory when unloading table A1.

```sql
UNLOAD A1 DELETE PERSISTENT MEMORY;
```

**Related Information**

Table Partitioning

M_CS_PARTITIONS System View - Deprecated [page 1833]

**4.10.1.162 UNSET PSE Statement (System Management)**

Removes the purpose for a PSE store.

**Syntax**

```sql
UNSET PSE <pse_name> PURPOSE <purpose>
```
Syntax Elements

**pse_name**
Specifies the name of the PSE store.

\(<\text{pse}\_\text{name}>\) ::= <identifier>

**purpose**
Specifies the purpose of the PSE.

\(<\text{purpose}>\) ::= 
  \(\text{JWT}\) 
  \(\text{SAML}\) 
  \(\text{SAP LOGON}\) 
  \(\text{SSL}\) 
  \(\text{X509}\) 
  \(\text{LDAP}\) 
  \(\text{SOLACE}\) 
  \(\text{DATABASE REPLICA}\) 
  \(\text{REMOTE SOURCE}\)

Description

Removes the assigned purpose for a PSE store along with any assigned purpose providers or hosts.

Permissions

If you are not the owner of the PSE store, then you must have the REFERENCES privilege on the PSE object. For the SSL purpose, you must also have the SSL ADMIN system privilege. For the SAML or JWT purposes, you must also have the USER ADMIN system privilege. For all other authentication purposes, you must also have the TRUST ADMIN privilege.

Examples

Remove the SSL purpose from the example_pse PSE store.

```
UNSET PSE example_pse PURPOSE SSL;
```

Related Information

PSE_PURPOSE_OBJECTS System View [page 1621]
4.10.1.163 UNSET [SESSION] Statement (Session Management)

Unsets a session variable for the current session.

Syntax

```
UNSET [SESSION] <key>
```

Syntax Elements

`key`

Specifies the session variable to unset.

```
<Key> ::= <string_literal>
```

Description

The list of session variables in use for the current session can be found by querying the M_SESSION_CONTEXT system view.

Example

Set the session variable MY_VAR to abc.

```
SET 'MY_VAR' = 'abc';
```

Select the variable MY_VAR from the current session.

```
SELECT SESSION_CONTEXT('MY_VAR') FROM DUMMY;
```
Unset the session variable `MY_VAR`.

```
UNSET 'MY_VAR';
```

### Related Information

- SET [SESSION] Statement (Session Management) [page 1171]
- M_SESSION_CONTEXT System View [page 2133]
- Session Variables [page 75]

#### 4.10.1.164 UNSET SYSTEM LICENSE ALL Statement (System Management)

Deletes all currently installed license keys.

⚠️ **Caution**

Running this command immediately locks down your SAP HANA database. You will then need a new license key before you can use the system further.

### Syntax

```
UNSET SYSTEM LICENSE ALL
```

### Permissions

You need the LICENSE ADMIN system privilege to execute this statement.

### Description

After using this command, the SAP HANA database is immediately locked down and requires a new, valid license key before it can be used further.
Example

Delete all currently installed license keys.

UNSET SYSTEM LICENSE ALL;

4.10.1.165 UPDATE Statement (Data Manipulation)

Changes the values of the records of a table.

Syntax

```
UPDATE [ <top_clause> ] <table_name> [ AS <alias_name> ]
  [ FOR PORTION_OF_APPLICATION_TIME <from_to_spec> ]
  [ <partition_restriction > ]
  [ <from_clause> ]
  <set_clause>
  [ WHERE <condition> ]
  [ <hint_clause> ]
```

Syntax Elements

**top_clause**
Limits the number of updated records (for example, for chunkwise updates).

```
<top_clause> ::= TOP <unsigned_integer>
```

*<unsigned_integer>* defines the number of records updated in one chunk.

**table_name**
Specifies the table where the UPDATE is performed, with the optional schema name.

```
<table_name> ::= [ [ <database_name>.]<schema_name>.]<identifier>
  <schema_name> ::= <unicode_name>
```

For linked database, *<database_name>* is the name of the remote source. *<identifier>* is the name of the table on the remote source.

**alias_name**
Specifies the alias that can be used to refer to the table defined by *<table_name>*.

```
(alias_name) ::= <identifier>
```

FOR PORTION OF APPLICATION_TIME
For use with application-time period tables, this clause specifies that the update be limited to the specified period of time between `<valid_from>` and `<valid_to>`, inclusively.

```sql
FOR PORTION OF APPLICATION_TIME <from_to_spec>
<from_to_spec> ::= FROM <valid_from> TO <valid_to>
```

`<valid_from>` and `<valid_to>` are date values found in the application-time period table specified for `<table_name>`.

**partition_restriction**

For a partitioned table, `<partition_restriction>` specifies the partition in which the target updated rows are located.

```sql
<partition_restriction> ::= PARTITION ( <partition_number> )
<partition_number> ::= <unsigned_integer>
```

If you do not specify a partition restriction, then the database checks all partitions for the update. But if there is partition restriction, only the rows in specified partition are updated.

**Limitations:**

- An UPDATE statement with a partition restriction is blocked if the following are true:
  - Partitioning key column update
  - UNIQUE constraint column update
  - PRIMARY KEY column update
- Updates that change the updated rows partition are not allowed.

**from_clause**

Specifies inputs such as tables to be used in the update statement.

```sql
<from clause> ::= FROM <table_name> [, <table_name>]...
<table_name> ::= <identifier>
```

The object to be updated can be specified as an alias in the `<from_clause>`.

`<from_clause>` is not supported in combination with FOR PORTION OF

**set_clause**

Specifies the column names and associated values to be set by the update statement.

```sql
<set_clause> ::= SET {<column_name> = <expression>
| ( <with_clause> <subquery> ) },...
```

For the definitions of the `<with_clause>` and `<subquery>`, see the SELECT statement. The `<with_clause>` can be used with column names, not with column lists.

**WHERE condition**

Specifies the conditions when the command should be performed.

```sql
<condition> ::= 
| <condition> OR <condition>
| <condition> AND <condition>
| NOT <condition>
| ( <condition> )
| <predicate>
| CURRENT OF <cursor>
```
WHERE CURRENT OF <cursor> specifies to perform the update at the current position in the cursor.

For more information on <predicate>, see the Predicates topic in this guide.

hint_clause

For information on hints, see the HINT clause in the SELECT statement.

Description

When this command is used with the FROM clause, the UPDATE FROM query must include the updated table name in the FROM clause. Or if an alias exists, the alias should exist in the FROM clause. Otherwise an error is returned.

If the WHERE condition is used and is true for a specific row, then the an update is performed on that row. If the WHERE clause is omitted, then the UPDATE command updates all records of a table.

While some database vendors do not support multiple updates to one record by an UPDATE FROM statement; SAP HANA does. However, the result can be non-deterministic since the order in which the updates are performed cannot always be guaranteed.

Examples

Create table T, and insert two rows into it.

```sql
CREATE TABLE T (KEY INT PRIMARY KEY, VAL INT);
INSERT INTO T VALUES (1, 1);
INSERT INTO T VALUES (2, 2);
```

Update the rows of table T if the condition in the WHERE clause is true.

```sql
UPDATE T SET VAL = VAL + 1 WHERE KEY = 1;
SELECT * FROM T;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

This example shows how to update all rows of the table by omitting the WHERE clause.

```sql
UPDATE T SET VAL = KEY + 10;
SELECT * FROM T;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
</tr>
</tbody>
</table>
Create table T2, and insert two rows into it.

```
CREATE ROW TABLE T2 (KEY INT PRIMARY KEY, VAR INT);
INSERT INTO T2 VALUES (1, 2);
INSERT INTO T2 VALUES (3, 6);
```

Update the values of table T by joining the target table T with table T2.

```
UPDATE T SET VAL = T2.VAR FROM T, T2 WHERE T.KEY = T2.KEY;
SELECT * FROM T;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
</tbody>
</table>

Update the table T with an aliased table in the FROM clause by joining the target table T with table T2 which also has an alias.

```
UPDATE T A SET VAL = B.VAR FROM T A, T2 B WHERE A.KEY = B.KEY;
SELECT * FROM T;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
</tbody>
</table>

The updated table should exist in the FROM clause.

If you update a table using the FROM clause like the example below, ambiguity of the target table in the FROM clause is allowed.

```
UPDATE T SET VAL = B.VAL FROM T A, T B WHERE A.KEY = B.KEY; --> error due to ambiguity of table T.
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
</tbody>
</table>

You can specify the partition that you want to update with partition restriction clause.

```
CREATE COLUMN TABLE PARTTAB1 (A INT, B INT, C INT) PARTITION BY RANGE(A) (PARTITION 0 <= VALUES <3, PARTITION OTHERS);
INSERT INTO PARTTAB1 VALUES (1,1,1);
INSERT INTO PARTTAB1 VALUES (2,2,2);
INSERT INTO PARTTAB1 VALUES (3,3,3);
```
Update a chunk of 100,000 records of the table testtab that contain 'XXX' in the request column and the value 0 in the updated column. For each row processed, the updated column is set to 1.

```
update top 100000 testtab set updated = 1 where request = 'XXX' and updated = 0
```

UPDATE a table with an array value construction by enumeration.

```
UPDATE T1 SET C1 = ARRAY ( 11, 12, 13, 14 ) WHERE ID = 1
```

UPDATE a table with an array value construction by query.

```
UPDATE T1 SET C1 = ARRAY( SELECT C1 FROM T0 ) WHERE ID = 1
```

The following statements provide an example of updating a single table using updatable cursor:

```
CREATE COLUMN TABLE employees (employee_id INTEGER, employee_name VARCHAR(30));
INSERT INTO employees VALUES (1, 'John');
INSERT INTO employees VALUES (20010, 'Sam');
INSERT INTO employees VALUES (21, 'Julie');
INSERT INTO employees VALUES (10005, 'Kate');
DO BEGIN
  DECLARE CURSOR cur FOR SELECT * FROM employees FOR UPDATE;
  FOR r AS cur DO
    IF r.employee_id < 10000 THEN
      UPDATE employees SET employee_id = employee_id + 10000
      WHERE CURRENT OF cur;
    ELSE
      DELETE FROM employees WHERE CURRENT OF cur;
    END IF;
  END FOR;
END;
```

The following statements provide an example of updating and deleting multiple tables (currently only COLUMN tables are supported) through an updatable cursor. In this case, you must specify columns of tables to be locked using FOR UPDATE OF within the cursor's SELECT statement. DML execution using updatable cursor is allowed just one time per row.

```
CREATE COLUMN TABLE employees (employee_id INTEGER, employee_name VARCHAR(30),
  department_id INTEGER);
INSERT INTO employees VALUES (1, 'John', 1);
INSERT INTO employees VALUES (2, 'Sam', 2);
INSERT INTO employees VALUES (3, 'Julie', 3);
INSERT INTO employees VALUES (4, 'Kate', 4);
CREATE COLUMN TABLE departments (department_id INTEGER, department_name VARCHAR(20));
INSERT INTO departments VALUES (1, 'Development');
INSERT INTO departments VALUES (2, 'Operation');
INSERT INTO departments VALUES (3, 'HR');
INSERT INTO departments VALUES (4, 'Security');
DO BEGIN
  DECLARE CURSOR cur FOR SELECT employees.employee_name,
    departments.department_name
  FROM employees, departments
  WHERE employees.department_id = departments.department_id
    FOR UPDATE OF employees.employee_id, departments.department_id;
  FOR r AS cur DO
    IF r.department_name = 'Development' THEN
```
UPDATE employees SET employee_id = employee_id + 10000, department_id = department_id + 100
WHERE CURRENT OF cur;
UPDATE departments SET department_id = department_id + 100
WHERE CURRENT OF cur;
ELSEIF r.department_name = 'HR' THEN
  DELETE FROM employees WHERE CURRENT OF cur;
  DELETE FROM departments WHERE CURRENT OF cur;
END IF;
END FOR;
END;

The following example shows how the UPDATE...FROM syntax supports multiple updates to one record, even though the result can be non-deterministic.

CREATE COLUMN TABLE T (KEY INT PRIMARY KEY, VAL INT);
INSERT INTO T VALUES (1, 1);
CREATE COLUMN TABLE T2 (KEY INT, VAR INT);
INSERT INTO T2 VALUES (1, 2);
INSERT INTO T2 VALUES (1, 3);
INSERT INTO T2 VALUES (1, 4);
INSERT INTO T2 VALUES (1, 5);
UPDATE T SET VAL = T2.VAR FROM T, T2 WHERE T.KEY = T2.KEY;
SELECT * FROM T;   ----> [(1,5)]

While the last update (1,5) is returned as the result, the sequence of updates isn't guaranteed so the result is non-deterministic.

In this example, suppose you have the following application-period time table, TAB01, that you want to update:

<table>
<thead>
<tr>
<th>ValidFrom</th>
<th>ValidTo</th>
<th>PKEY</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-08-01</td>
<td>2018-08-15</td>
<td>1</td>
<td>9494</td>
</tr>
<tr>
<td>2018-08-15</td>
<td>2018-08-31</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>2018-08-31</td>
<td>2018-09-10</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>2018-09-10</td>
<td>9999-12-31</td>
<td>1</td>
<td>40</td>
</tr>
</tbody>
</table>

When you execute the following UPDATE statement, the values are updated according to the description in the Comment column of table that follows:

UPDATE TAB01 FOR PORTION OF '2018-08-20' AND '2018-10-20' SET VALUE=42 WHERE PKEY=1;

<table>
<thead>
<tr>
<th>ValidFrom</th>
<th>ValidTo</th>
<th>PKEY</th>
<th>VALUE</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-08-01</td>
<td>2018-08-15</td>
<td>1</td>
<td>9394</td>
<td>Not modified (outside of portion interval)</td>
</tr>
<tr>
<td>2018-08-15</td>
<td>2018-08-20</td>
<td>1</td>
<td>20</td>
<td>Newly inserted record (case partial overlap)</td>
</tr>
<tr>
<td>2018-08-20</td>
<td>2018-08-31</td>
<td>1</td>
<td>42</td>
<td>Valid from and Value were updated (case partial overlap)</td>
</tr>
<tr>
<td>2018-08-31</td>
<td>2018-09-10</td>
<td>1</td>
<td>42</td>
<td>Only Value was updated (case full overlap)</td>
</tr>
</tbody>
</table>
Related Information

Introduction to SQL [page 27]
Expressions [page 56]
SELECT Statement (Data Manipulation) [page 1104]
Predicates [page 61]

4.10.1.166 UPSERT Statement (Data Manipulation)

Updates rows in a table or inserts new rows. This UPSERT and REPLACE statements are synonymous and have similar syntax and purpose.

Syntax

```sql
UPSERT <table_name> [ <partition_restriction> ] [ <column_list_clause> ]
{       <value_list_clause> [ WHERE <condition> | WITH PRIMARY KEY ]
       | <subquery>
}
```

Syntax Elements

table_name

Specifies the table to update, with the optional schema name.

```sql
<table_name> ::= [ [ <database_name>.]<schema_name>.]<identifier>
        <schema_name> ::= <unicode_name>
```

For linked database, `<database_name>` is the name of the remote source. `<identifier>` is the name of the table on the remote source.

column_list_clause
Specifies a list of column identifiers, ordered in the order of values in the `<value_list_clause>` or `<subquery>`.

```
<column_list_clause> ::= ( <column_name> [,..] )
<column_name> ::= <identifier>
```

**partition_restriction**

For a partitioned table, `<partition_restriction>` specifies the partition in which the target rows are located.

```
<partition_restriction> ::= PARTITION ( <<partition_number> )
<partition_number> ::= <unsigned_integer>
```

If you do not specify a partition restriction, then the database checks all partitions for the update. But if there is a partition restriction, then only the rows in the specified partition are updated. Use the TABLE_PARTITIONS system view to determine the partition number.

The `<partition_restriction>` clause is supported for multistore tables but not extended store tables.

**value_list_clause**

Specifies a list of values, or expressions evaluating to values, to be inserted or updated into the table.

```
<value_list_clause> ::= VALUES ( <expression> [,..] )
```

**WHERE condition**

Specifies the conditions where the command should be performed.

```
<condition> ::= 
    <condition> OR <condition>
    | <condition> AND <condition>
    | NOT <condition>
    | ( <condition> )
    | <predicate>
```

**WITH PRIMARY KEY**

Uses the primary key value in the `<value_list_clause>` to define the row to act upon.

**subquery**

For information on subqueries, see the SELECT statement.

---

### Description

When this command is used without a subquery, it functions in a similar way to the UPDATE statement. The difference with this command is that when the WHERE clause condition is false it inserts a new record into the table.

When this command is used with a table that has a primary key, the primary key column must be included in the column list. Columns defined with NOT NULL and without a default specification also have to be included in the column list. Columns that are not specified are filled with a default value or NULL.

The UPSERT (or REPLACE) statement with a subquery operates like the INSERT statement. The exception with this command is that, if an existing row in the table has the same primary key value as a new row, the row is updated with the returned record from the subquery. If the table does not have a primary key, then the
command functions in an equivalent way to an INSERT statement as an index cannot be used to check for row duplication.

**Example**

Create table T.

```
CREATE ROW TABLE T (KEY INT PRIMARY KEY, VAL INT);
```

Insert a new value.

```
UPSERT T VALUES (1, 1);
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Insert a new value if the condition in the WHERE clause is false or update the current row values if WHERE evaluates to true.

```
UPSERT T VALUES (2, 2) WHERE KEY = 2;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Update the table row where KEY is equal to 1.

```
UPSERT T VALUES (1, 9) WHERE KEY = 1;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Use the "WITH PRIMARY KEY" keyword to update the table using the primary key value in the VALUES clause.

```
UPSERT T VALUES (1, 8) WITH PRIMARY KEY;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>
Insert values using a subquery.

```
UPSERT T SELECT KEY + 2, VAL FROM T;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

UPSERT a table with an array value construction by enumeration.

```
UPSERT T1 VALUES ( 1, ARRAY ( 21, 22, 23, 24 ) ) WHERE ID = 1;
```

UPSERT a table with an array value construction by query.

```
UPSERT T1 VALUES ( 1, ARRAY ( SELECT C1 FROM T0 ) ) WHERE ID = 1;
```

UPSERT values into partition 1 of table T1 and then upsert values from table T1 to partition 1 in table T2.

```
CREATE COLUMN TABLE T1 (a int, b int) PARTITION BY RANGE (a)
   (PARTITION 0 <= VALUES < 10, PARTITION OTHERS);
CREATE COLUMN TABLE T2 (a int, b int) PARTITION BY RANGE (a)
   (PARTITION 0 <= VALUES < 10, PARTITION OTHERS);
INSERT INTO T1 PARTITION(1) VALUES(1,50);
UPSERT T1 PARTITION(1) VALUES(1,20);
UPSERT T2 PARTITION(1) SELECT * FROM T1;
Select * from T2;
```

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>

**Related Information**

- SELECT Statement (Data Manipulation) [page 1104]
- REPLACE Statement (Data Manipulation) [page 1096]
- UPDATE Statement (Data Manipulation) [page 1185]
- TABLE_PARTITIONS System View [page 1681]
4.10.1.167 VALIDATE LDAP PROVIDER Statement (Access Control)

Validates an LDAP provider configuration and LDAP authentication and authorization for users of that LDAP provider.

Syntax

```
VALIDATE LDAP PROVIDER <ldap_provider_name> [ <validate_clause> ]
```

Syntax Elements

- **ldap_provider_name**
  The identifier for a valid LDAP provider previously created using the CREATE LDAP PROVIDER statement.

  ```
  <ldap_provider_name> ::= <unicode_name>
  ```

- **validate_clause**
  Specifies the type of validation to perform.

  ```
  <validate_clause> ::=   CHECK USER <user_name> [ PASSWORD <password_string_literal> | NO AUTHORIZATION CHECK ]    | CHECK USER CREATION FOR LDAP USER <user_name>
  ```

If this clause is not specified, then the statement is successful if SAP HANA can connect and bind to the named LDAP server using the credential of the access lookup account as configured in the LDAP provider.

- **CHECK USER user_name**
  Checks LDAP authorization for the user.

  SAP HANA connects and binds to the named LDAP server and then performs the LDAP search using the USER LOOKUP URL specified for the LDAP provider after replacing ‘*’ with the specified `<user_name>` and after replacing the `<attributes>` element of the URL with the attributes specified by DN and MEMBER_OF (if NESTED GROUP LOOKUP URL is not defined). If NESTED GROUP LOOKUP URL is defined, then to obtain the user’s LDAP group information, SAP HANA performs another LDAP search using NESTED GROUP LOOKUP URL after replacing ‘*’ with the DN of the user obtained from the USER LOOKUP URL search. This statement is successful if the user has at least one role with a mapping to LDAP groups obtained from a USER LOOKUP URL or NESTED GROUP LOOKUP URL search.

- **CHECK USER user_name PASSWORD password_string_literal**
  Checks LDAP authentication for the user.

  SAP HANA connects and binds to the named LDAP server and then performs the LDAP search using the USER LOOKUP URL specified for the LDAP provider after replacing ‘*’ with the specified
<user_name> and after replacing the <attributes> element of the URL with the attribute specified by DN. The search is performed and if no errors occur, then the result value for the DN and the value of password are used to again connect to the LDAP server. When the connection with DN and password is successful, this is successful authentication of the user.

CHECK USER username NO AUTHORIZATION CHECK

Checks whether the user exists in LDAP; that is, neither an LDAP authorization nor an LDAP authentication check is done.

SAP HANA connects and binds to the named LDAP server and then performs the LDAP search using the USER LOOKUP URL specified for the LDAP provider after replacing '*' with the specified <user_name> and after replacing the <attributes> element of the URL with the attribute specified by DN. The search is performed and if no errors occur, then this successfully verifies that the user exists on the LDAP server. This is useful when authorization checking is not to be done or the user’s password is unknown, since it validates the user existence on the LDAP server and validates the values specified in the LDAP provider.

CHECK USER CREATION FOR LDAP USER user_name

Checks the LDAP configuration required for automatic user creation with LDAP authentication, including verifying the LDAP authorization for the specified user.

SAP HANA connects and binds to the named LDAP server and then performs the LDAP search using the USER LOOKUP URL specified for the LDAP provider after replacing '*' with the specified <user_name> and after replacing the <attributes> element of the URL with the attributes specified by DN and MEMBER_OF. The search is performed and if no errors occur, then the result values for the DN and MEMBER_OF attributes are then used to see if the user has at least one role with a mapping to a member LDAP Group and a user with that name doesn’t already exist, assuming NESTED GROUP LOOKUP URL is not defined. If NESTED GROUP LOOKUP URL is defined, then to obtain the user’s LDAP group information, SAP HANA performs another LDAP search using NESTED GROUP LOOKUP URL after replacing '*' with the DN of the user obtained from the USER LOOKUP URL search. The statement is successful if the user has at least one role with a mapping to LDAP groups obtained from a USER LOOKUP URL or NESTED GROUP LOOKUP URL search, and if automatic user creation is configured for the LDAP provider and a user with that name doesn’t already exist in SAP HANA.

Description

Validates an LDAP provider configuration and LDAP authentication and authorization for users of that LDAP provider.

Permissions

Only database users with the LDAP ADMIN system privilege can execute the VALIDATE LDAP PROVIDER statement.
Examples

EXAMPLE 1: The following statement causes SAP HANA to perform the validation steps listed after the statement.

```sql
VALIDATE LDAP PROVIDER my_ldap_provider CHECK USER johnd;
```

1. Check that USER LOOKUP URL is a valid URL, and that either ATTRIBUTE MEMBER_OF is defined or NESTED GROUP LOOKUP URL is defined and is a valid URL.
2. Bind to the LDAP server specified in the USER LOOKUP URL using the access account credentials specified in the provider.
3. Search for user 'johnd' using the USER LOOKUP URL to obtain the user's distinguished name from the LDAP server.
4. Obtain the user's LDAP group membership from the LDAP server. Find matching roles based on the user's LDAP groups and mapping entries in the SAP HANA server between SAP HANA roles in LDAP groups.

A successful return validates the configuration of the LDAP provider for LDAP authorization, and that the user exists in LDAP server and that the user has at least one role in SAP HANA that can be granted to the user based on user's LDAP group membership in LDAP server and role to LDAP group mapping in SAP HANA.

EXAMPLE 2: The following statement causes SAP HANA to perform the validation steps listed after the statement.

```sql
VALIDATE LDAP PROVIDER my_ldap_provider CHECK USER testuser1 PASSWORD 'Sap123456';
```

1. Check that the USER LOOKUP URL is a valid LDAP URL.
2. Bind to the LDAP server specified in the USER LOOKUP URL using the access account credentials specified in the provider.
3. Search for user 'testuser1' using the USER LOOKUP URL to obtain user's distinguished name from the LDAP server.
4. Bind to the LDAP server specified in the USER LOOKUP URL using the user's distinguished name and the specified password.

A successful return validates the configuration of the LDAP provider for LDAP authentication and that the user exists in the LDAP server and can successfully bind to the LDAP server using the specified password.

EXAMPLE 3: The following statement causes SAP HANA to perform the validation steps listed after the statement.

```sql
VALIDATE LDAP PROVIDER my_ldap_provider CHECK USER testuser2 NO AUTHORIZATION CHECK;
```

1. Check that the USER LOOKUP URL is a valid LDAP URL.
2. Bind to the LDAP server specified in USER LOOKUP URL using the access account credentials specified in the provider.
3. Search for user 'testuser2' using the USER LOOKUP URL to obtain user's distinguished name from the LDAP server.

A successful return validates the configuration of the LDAP provider and that the user exists in the LDAP server.
4.10.1.168  VALIDATE USER Statement (Access Control)

Validates the credentials of a user in the current database.

Syntax

```
VALIDATE USER { <user_name> PASSWORD <password> | WITH ASSERTION <assertion> }
```

Syntax Elements

- user_name
  
  Specifies the identifier for the credentials being validated.

- password
  
  Specifies the password for the credentials being validated.

- assertion
  
  Specifies a valid ticket for a single-sign on service such as SAML, JWT, SAPLOGON.

Description

VALIDATE USER can be used to validate the credentials for a user in the current database without causing a login using the credentials.

When validation fails, an error is returned indicating that authentication failed. The error does not indicate whether the failure is because `<user_name>` does not exist or because the specified `<password>` does not match the current password for `<user_name>`.

After successful validation, the user name can be found in the session variable `VALIDATE_USER_NAME`. 
Examples

The following example validates the credentials of user JSmith and returns a message indicating that the statement executed successfully (JSmith is a valid user):

```sql
CREATE USER JSmith PASSWORD shg8475ghh11;
VALIDATE USER JSmith PASSWORD shg8475ghh11;
```

The following example returns an error indicating that validation of user T12345 failed. Although the error does not indicate whether it failed because the user name doesn’t exist or whether the password is incorrect, you can see from the example that the reason is because the password specified in the VALIDATE USER statement doesn’t match the user’s password:

```sql
CREATE USER T12345 PASSWORD suelg75sjh;
VALIDATE USER T12345 PASSWORD shgi33shi;
```

The following example returns an error indicating that validation failed (because the user BJones has not been created in this example):

```sql
VALIDATE USER BJones PASSWORD 34875hgj75;
```

The following example uses a single-sign on ticket to validate a user, and returns an error message indicating that authentication failed (because the single-sign on ticket in this case is fictitious):

```sql
VALIDATE USER WITH ASSERTION
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDCeFxjoY2geFONFh7HgQ';
```

4.10.2 Access Control Statements

Access control statements enable database administrators to create, alter, and drop access to the SAP HANA database.

Supported Statements

- ALTER AUDIT POLICY Statement (Access Control) [page 431]
- ALTER CREDENTIAL Statement (Access Control) [page 439]
- ALTER JWT PROVIDER Statement (Access Control) [page 455]
- ALTER LDAP PROVIDER Statement (Access Control) [page 458]
- ALTER REMOTE SOURCE Statement (Access Control) [page 471]
- ALTER ROLE Statement (Access Control) [page 475]
- ALTER SAML PROVIDER Statement (Access Control) [page 476]
- ALTER STRUCTURED PRIVILEGE Statement (Access Control) [page 491]
- ALTER USER Statement (Access Control) [page 654]
- ALTER USERGROUP Statement (Access Control) [page 665]
4.10.3 Backup and Recovery Statements

These statements allow users to perform backup and recovery operations.

Supported Statements

- BACKUP CANCEL Statement (Backup and Recovery) [page 688]
- BACKUP CATALOG DELETE Statement (Backup and Recovery) [page 690]
- BACKUP CATALOG RETAINED Statement (Backup and Recovery) [page 693]
- BACKUP CHECK Statement (Backup and Recovery) [page 695]
- BACKUP CHECK ACCESS Statement (Backup and Recovery) [page 696]
- BACKUP DATA Statement (Backup and Recovery) [page 699]
- BACKUP DATA CLOSE SNAPSHOT Statement (Backup and Recovery) [page 704]
- BACKUP DATA CREATE SNAPSHOT Statement (Backup and Recovery) [page 706]
4.10.4 Data Definition Statements

Data definition statements define structures in the SAP HANA database.

Supported Statements

- ALTER FULLTEXT INDEX Statement (Data Definition) [page 445]
- ALTER INDEX Statement (Data Definition) [page 452]
- ALTER SEQUENCE Statement (Data Definition) [page 479]
- ALTER STATISTICS Statement (Data Definition) [page 484]
- ALTER SCHEDULER JOB Statement (Data Definition) [page 482]
- ALTER TABLE Statement (Data Definition) [page 589]
- ALTER VIEW Statement (Data Definition) [page 667]
- ALTER VIRTUAL TABLE Statement (Data Definition) [page 676]
- ANNOTATE Statement (Data Definition) [page 686]
- COMMENT ON Statement (Data Definition) [page 718]
- CREATE FULLTEXT INDEX Statement (Data Definition) [page 748]
- CREATE GRAPH WORKSPACE Statement (Data Definition) [page 760]
- CREATE INDEX Statement (Data Definition) [page 762]
- CREATE FUZZY SEARCH INDEX Statement (Data Definition) [page 759]
- CREATE PROJECTION VIEW Statement (Data Definition) [page 794]
- CREATE SCHEDULER JOB Statement (Data Definition) [page 806]
- CREATE SCHEMA Statement (Data Definition) [page 807]
- CREATE SCHEMA SYNONYM Statement (Data Definition) [page 808]
- CREATE SEQUENCE Statement (Data Definition) [page 810]
- CREATE STATISTICS Statement (Data Definition) [page 815]
- CREATE SYNONYM Statement (Data Definition) [page 823]
- CREATE TABLE Statement (Data Definition) [page 825]
- CREATE TRIGGER Statement (Data Definition) [page 876]
- CREATE VIEW Statement (Data Definition) [page 904]
- CREATE VIRTUAL TABLE Statement (Data Definition) [page 925]
- DROP FULLTEXT INDEX Statement (Data Definition) [page 955]
4.10.5 Data Import Export Statements

The following statements import and export data to and from the SAP HANA database.

Supported Statements

- EXPORT INTO Statement (Data Import Export) [page 1007]
- EXPORT Statement (Data Import Export) [page 997]
- IMPORT FROM Statement (Data Import Export) [page 1038]
- IMPORT SCAN Statement (Data Import Export) [page 1047]
- IMPORT Statement (Data Import Export) [page 1029]
4.10.6 Data Manipulation Statements

The following statements enable you to manipulate data within the SAP HANA database.

**Supported Statements**

- DELETE Statement (Data Manipulation) [page 938]
- EXPLAIN PLAN Statement (Data Manipulation) [page 992]
- INSERT Statement (Data Manipulation) [page 1050]
- LOAD Statement (Data Manipulation) [page 1054]
- MERGE DELTA Statement (Data Manipulation) [page 1058]
- MERGE INTO Statement (Data Manipulation) [page 1060]
- REPLACE Statement (Data Manipulation) [page 1096]
- SELECT Statement (Data Manipulation) [page 1104]
- TRUNCATE TABLE Statement (Data Manipulation) [page 1178]
- UNLOAD Statement (Data Manipulation) [page 1179]
- UPDATE Statement (Data Manipulation) [page 1185]
- UPSERT Statement (Data Manipulation) [page 1191]

4.10.7 Client-side Encryption Statements

Client-side encryption statements enable database administrators to administer client-side encrypted data in the SAP HANA database.

**Supported Statements**

- ALTER CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 435]
- ALTER CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 437]
- CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 741]
- CREATE CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 743]
- DROP CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 949]
- DROP CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 950]
4.10.8 JSON Document Store Statements

There are several JSON Document Store-specific variants of common DDL and DML SQL statements.

Supported Statements

To see the list of statements and how to use them to create collection tables and manage JSON data, see the SAP HANA JSON Document Store Guide.

Related Information

SAP HANA JSON Document Store Guide
M_TABLES System View [page 2200]
TABLES System View [page 1669]

4.10.9 Procedural Statements

Procedural statements manage system and user-defined procedures for the SAP HANA database.

Supported Statements

- ALTER FUNCTION Statement (Procedural) [page 447]
- ALTER PROCEDURE Statement (Procedural) [page 463]
- CALL Statement (Procedural) [page 715]
- CREATE FUNCTION Statement (Procedural) [page 752]
- CREATE PROCEDURE Statement (Procedural) [page 776]
- CREATE TYPE Statement (Procedural) [page 892]
- CREATE VIRTUAL FUNCTION Statement (Procedural) [page 917]
- CREATE VIRTUAL PROCEDURE Statement (Procedural) [page 923]
- DO BEGIN...END Statement (Procedural) [page 942]
- DROP FUNCTION Statement (Procedural) [page 956]
- DROP PROCEDURE Statement (Procedural) [page 963]
- DROP TYPE Statement (Procedural) [page 983]
4.10.10 Session Management Statements

The following SQL statements manage database sessions.

**Supported Statements**

- CONNECT Statement (Session Management) [page 721]
- SET [SESSION] Statement (Session Management) [page 1171]
- SET HISTORY SESSION Statement (Session Management) [page 1165]
- SET SCHEMA Statement (Session Management) [page 1170]
- UNSET [SESSION] Statement (Session Management) [page 1183]

**Related Information**

Session Variables [page 75]

4.10.11 System Management Statements

The following statements enable you to manage system configuration settings in your SAP HANA database.

**Supported Statements**

- ALTER PSE Statement (System Management) [page 468]
- ALTER SYSTEM {ADD | ALTER | REMOVE} STATEMENT HINT Statement (System Management) [page 493]
- ALTER SYSTEM {ADD | ACTIVATE | UPDATE | DROP} KEY MANAGEMENT CONFIGURATION Statement (System Management) [page 496]
- ALTER SYSTEM {ADD | REMOVE} ABSTRACT SQL PLAN FILTER (System Management) [page 497]
- ALTER SYSTEM ALTER CONFIGURATION Statement (System Management) [page 500]
- ALTER SYSTEM ALTER DATAVOLUME ADD PARTITION Statement (System Management) [page 503]
- ALTER SYSTEM ALTER DATAVOLUME DROP PARTITION Statement (System Management) [page 504]
- ALTER SYSTEM ALTER SESSION SET Statement (System Management) [page 506]
- ALTER SYSTEM ALTER SESSION UNSET Statement (System Management) [page 507]
- ALTER SYSTEM ALTER TABLE PLACEMENT Statement (System Management) [page 508]
- ALTER SYSTEM APPLICATION ENCRYPTION Statement (System Management) [page 512]
- ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
4.10.12 Tenant Database Management Statements

The following SQL statements manage SAP HANA tenant databases.

Supported Statements

- ALTER DATABASE Statement (Tenant Database Management) [page 440]
- ALTER SYSTEM START DATABASE Statement (Tenant Database Management) [page 583]
- ALTER SYSTEM STOP DATABASE Statement (Tenant Database Management) [page 584]
- CREATE DATABASE Statement (Tenant Database Management) [page 746]
- DROP DATABASE Statement (Tenant Database Management) [page 954]
- RENAME DATABASE Statement (Tenant Database Management) [page 1090]
4.10.13 Transaction Management Statements

The following SQL statements manage transactions in the SAP HANA database.

Supported Statements

- COMMIT Statement (Transaction Management) [page 720]
- LOCK TABLE Statement (Transaction Management) [page 1056]
- RELEASE SAVEPOINT Statement (Transaction Management) [page 1087]
- ROLLBACK Statement (Transaction Management) [page 1101]
- ROLLBACK TO SAVEPOINT Statement (Transaction Management) [page 1102]
- SAVEPOINT Statement (Transaction Management) [page 1103]
- SET TRANSACTION Statement (Transaction Management) [page 1173]
- SET TRANSACTION AUTOCOMMIT DDL Statement (Transaction Management) [page 1176]

4.10.14 Workload Management Statements

The following SQL statements manage workload classes and mappings.

Supported Statements

- ALTER WORKLOAD CLASS Statement (Workload Management) [page 678]
- ALTER WORKLOAD MAPPING Statement (Workload Management) [page 681]
- CREATE WORKLOAD CLASS Statement (Workload Management) [page 929]
- CREATE WORKLOAD MAPPING Statement (Workload Management) [page 934]
- DROP WORKLOAD CLASS Statement (Workload Management) [page 989]
- DROP WORKLOAD MAPPING Statement (Workload Management) [page 990]

4.11 System Limitations

Limitations to take into consideration when administering an SAP HANA database.

Aside from the table below, most system limits can also be viewed by querying the M_SYSTEM_LIMITS system view (SELECT * FROM M_SYSTEM_LIMITS; ). However, your values might differ depending on the hardware and software configuration your system uses.
For details of sizing and limitations regarding HDI containers, refer to the topic SAP HDI Sizing and Limitations in the guide SAP HANA Deployment Infrastructure (HDI) Reference for SAP HANA Platform (see Related Information below).

<table>
<thead>
<tr>
<th>Limitation Area</th>
<th>Limit</th>
<th>M_SYSTEM_LIMITS view name for the limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database size limit</td>
<td>Row Store: 1.945 GB</td>
<td>MAXIMUM_SIZE_OF_ROW_STORE</td>
</tr>
<tr>
<td></td>
<td>Column Store: Dependent on size of physical memory</td>
<td></td>
</tr>
<tr>
<td>Number of locks</td>
<td>Unlimited for record locks, 16,383 for table locks</td>
<td>MAXIMUM_NUMBER_OF_TABLE_LOCKS</td>
</tr>
<tr>
<td>Number of sessions</td>
<td>65,536</td>
<td>MAXIMUM_NUMBER_OF_SESSIONS</td>
</tr>
</tbody>
</table>

**Schema Limitations**

<table>
<thead>
<tr>
<th>Number of schemas per SAP HANA instance</th>
<th>Maximum value of BIGINT data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier length</td>
<td>127 bytes</td>
</tr>
<tr>
<td>Length of an alias name</td>
<td>128 characters</td>
</tr>
<tr>
<td>Table name length</td>
<td>Same as Identifier length</td>
</tr>
<tr>
<td>Column name length</td>
<td>Same as Identifier length</td>
</tr>
<tr>
<td>Length of a string literal</td>
<td>8 MB</td>
</tr>
<tr>
<td>Number of hex characters in a binary literal</td>
<td>8,192 Bytes</td>
</tr>
</tbody>
</table>

**Tables and View Limitations**

<p>| Number of columns in a table           | 64,000                                                            | MAXIMUM_NUMBER_OF_COLUMNS_IN_TABLE                                |
|                                        | This limit can vary based on context, for example, in the context of virtual tables, SAP HANA may be limited by the capabilities of the remote system and the limit of the other DBMS may apply instead. In cases such as this, the limit that is met first becomes the actual limit. |
| Number of columns in a row table       | 1,000                                                             | MAXIMUM_NUMBER_OF_COLUMNS_IN_ROW_TABLE                            |</p>
<table>
<thead>
<tr>
<th><strong>Number of columns in a view</strong></th>
<th>64,000</th>
<th><strong>MAXIMUM_NUMBER_OF_COLUMNS_IN_VIEW</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of rows in each table</strong></td>
<td>Limited by storage size RS: 1,945 GB/sizeof(row), CS: 2,100,000,000 * number of partitions</td>
<td></td>
</tr>
<tr>
<td><strong>Length of a row</strong></td>
<td>Limited by RS storage size (1,945 GB per index server)</td>
<td></td>
</tr>
<tr>
<td><strong>Size of a non-partitioned table</strong></td>
<td>Limited by RS storage size (1,945 GB per index server)</td>
<td></td>
</tr>
<tr>
<td><strong>Number of partitions in a CS table</strong></td>
<td>16,000</td>
<td><strong>MAXIMUM_NUMBER_OF_PARTITIONS_IN_CSTABLE</strong></td>
</tr>
<tr>
<td><strong>Number of triggers per table per DML statement</strong></td>
<td>1,024</td>
<td><strong>MAXIMUM_NUMBER_OF_TRIGGERS_PER_TABLE_PER_DML</strong></td>
</tr>
<tr>
<td><strong>Number of records per (non-partitioned) table</strong></td>
<td>2,100,000,000</td>
<td></td>
</tr>
</tbody>
</table>

**Indexes and Constraints**

<table>
<thead>
<tr>
<th><strong>Number of indexes for a table</strong></th>
<th>1,023</th>
<th><strong>MAXIMUM_NUMBER_OF_INDEXES_IN_TABLE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of primary key columns in each table</strong></td>
<td>16</td>
<td><strong>MAXIMUM_NUMBER_OF_COLUMNS_IN_PRIMARY_KEY</strong></td>
</tr>
<tr>
<td><strong>Number of primary key columns in each column store table</strong></td>
<td>1,000</td>
<td><strong>MAXIMUM_NUMBER_OF_COLUMNS_IN_PRIMARY_KEY_IN_COLUMN_TABLE</strong></td>
</tr>
<tr>
<td><strong>Number of columns in an index</strong></td>
<td>16</td>
<td><strong>MAXIMUM_NUMBER_OF_COLUMNS_IN_INDEX</strong></td>
</tr>
<tr>
<td><strong>Number of columns in a UNIQUE constraint</strong></td>
<td>16</td>
<td><strong>MAXIMUM_NUMBER_OF_COLUMNS_IN_UNIQUE_CONSTRAINT</strong></td>
</tr>
<tr>
<td><strong>Size of sum of primary key, index, UNIQUE constraint</strong></td>
<td>16,384 Bytes</td>
<td><strong>MAXIMUM_SIZE_OF_KEY_IN_INDEX</strong></td>
</tr>
<tr>
<td><strong>Number of indexes in row store</strong></td>
<td>256,000</td>
<td></td>
</tr>
</tbody>
</table>

**SQL**
<table>
<thead>
<tr>
<th>Feature</th>
<th>Limit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of an SQL statement</td>
<td>2,147,483,648 B</td>
<td>Maximum length of SQL statement</td>
</tr>
<tr>
<td>Depth of SQL view nesting</td>
<td>128</td>
<td>Maximum depth of SQL view nesting</td>
</tr>
<tr>
<td>Maximum depth of SQL parse tree</td>
<td>0 (unlimited)</td>
<td>This limitation does not show in M_SYSTEM_LIMITS unless a limit is configured to something other than 0 (no limit) using the max_parse_tree_depth parameter in indexerver.ini.</td>
</tr>
<tr>
<td>Maximum depth of joins in a statement</td>
<td>0</td>
<td>Maximum depth of joins in a statement. This limitation does not show in M_SYSTEM_LIMITS unless a limit is configured to something other than 0 (no limit) using the max_join_depth parameter in indexerver.ini.</td>
</tr>
<tr>
<td>Number of columns in an ORDER BY</td>
<td>65,535</td>
<td>Maximum number of columns in ORDER BY</td>
</tr>
<tr>
<td>Number of columns in a GROUP BY</td>
<td>65,535</td>
<td>Maximum number of columns in GROUP BY</td>
</tr>
<tr>
<td>Number of elements in IN predicates</td>
<td>65,535</td>
<td>Maximum number of elements in IN predicates</td>
</tr>
<tr>
<td>Number of elements in SELECT clause</td>
<td>65,535</td>
<td>Maximum number of output columns in SELECT clause</td>
</tr>
<tr>
<td>Number of tables in a statement</td>
<td>0</td>
<td>Maximum number of tables in a statement. This limitation does not show in M_SYSTEM_LIMITS unless a limit is configured to something other than 0 (no limit) using the max_table_count_in_statement parameter in indexerver.ini.</td>
</tr>
<tr>
<td>LOB Limitations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum size of an in-memory LOB for a column store table</td>
<td>2 GB</td>
<td>Maximum size of in-memory LOB for a column store table</td>
</tr>
</tbody>
</table>
### Maximum size of an in-memory LOB for a row store table

2 GB

**MAXIMUM_SIZE_OF_MEMORY_LOB_IN_ROW_STORE**

### Maximum size of a packed LOB

1,013,760 bytes

**MAXIMUM_SIZE_OF_PACKED_LOB**

### Maximum size of a LOB on disk

4,294,967,295 bytes

**MAXIMUM_SIZE_OF_DISK_LOB**

### Procedures

#### Size of all stored procedures

1,945 GB

**MAXIMUM_SIZE_OF_ALL_STORED_PROCEDURES**

#### Size of a procedure definition

2 GB

**MAXIMUM_SIZE_OF_PROCEDURE_DEFINITION**

### Related Information

- **M_SYSTEM_LIMITS** System View [page 2192]
- **SAP HDI Sizing and Limitations**

### 4.12 SQL Error Codes

Each SAP HANA error is identified by a numeric error code and has a short descriptive text.

The **M_ERROR_CODES** system view shows the basic information about the error codes, execute the following query to get a complete list of codes and their descriptions:

```sql
SELECT * FROM M_ERROR_CODES ORDER BY CODE ASC;
```

The **SAP HANA Troubleshooting and Performance Analysis Guide** has a section on Error Codes including a reference topic with links to additional information for many error codes (link in Related Information).

The following table lists error codes and their descriptions for the SAP HANA database:

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WRN_GENERAL</td>
<td>General warning</td>
</tr>
<tr>
<td>2</td>
<td>ERR_GENERAL</td>
<td>General error</td>
</tr>
<tr>
<td>3</td>
<td>FATAL_GENERAL</td>
<td>Fatal error</td>
</tr>
<tr>
<td>4</td>
<td>FATAL_OUT_OF_MEMORY</td>
<td>Cannot allocate enough memory</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>5</td>
<td>ERR_INIT</td>
<td>Initialization error</td>
</tr>
<tr>
<td>6</td>
<td>ERR_DATA</td>
<td>Invalid data</td>
</tr>
<tr>
<td>7</td>
<td>ERR_FEATURE_NOT_SUPPORTED</td>
<td>Feature not supported</td>
</tr>
<tr>
<td>8</td>
<td>ERR_INV_ARGUMENT</td>
<td>Invalid argument</td>
</tr>
<tr>
<td>9</td>
<td>ERR_INDEX_OUT_OF_BOUNDS</td>
<td>Index out of bounds</td>
</tr>
<tr>
<td>10</td>
<td>ERR_AUTHENTICATION_FAILED</td>
<td>Authentication failed</td>
</tr>
<tr>
<td>11</td>
<td>ERR_INV_STATE</td>
<td>Invalid state</td>
</tr>
<tr>
<td>12</td>
<td>ERR_FILE_OPEN_FAILED</td>
<td>Cannot open file</td>
</tr>
<tr>
<td>13</td>
<td>ERR_FILE_CREATE_WRITE_FAILED</td>
<td>Cannot create/write file</td>
</tr>
<tr>
<td>14</td>
<td>ERR_DISK_SPACE_SHORTAGE</td>
<td>Cannot allocate enough disk space</td>
</tr>
<tr>
<td>15</td>
<td>ERR_FILE_NOT_FOUND</td>
<td>Cannot find file</td>
</tr>
<tr>
<td>16</td>
<td>ERR_RETRY_STATEMENT</td>
<td>Statement retry</td>
</tr>
<tr>
<td>17</td>
<td>ERR_CATA_VER_MISMATCH</td>
<td>Metadata schema version incompatible between database and executable file</td>
</tr>
<tr>
<td>18</td>
<td>ERR_SERVICE_SHUTDOWN</td>
<td>Service shutting down</td>
</tr>
<tr>
<td>19</td>
<td>ERR_INV_LICENSE</td>
<td>Invalid license</td>
</tr>
<tr>
<td>20</td>
<td>ERR_CON_OUTSIDE_VALIDITY_PERIOD</td>
<td>Connect attempt outside user’s validity period</td>
</tr>
<tr>
<td>21</td>
<td>ERR_PERSISTENCE</td>
<td>Persistence error</td>
</tr>
<tr>
<td>128</td>
<td>ERR_TX</td>
<td>Transaction error</td>
</tr>
<tr>
<td>129</td>
<td>ERR_TX_ROLLBACK</td>
<td>Transaction rolled back by an internal error</td>
</tr>
<tr>
<td>130</td>
<td>ERR_TX_ROLLBACK_INTEGRITY</td>
<td>Transaction rolled back by integrity constraint violation</td>
</tr>
<tr>
<td>131</td>
<td>ERR_TX_ROLLBACK_LOCK_TIMEOUT</td>
<td>Transaction rolled back by lock wait timeout</td>
</tr>
<tr>
<td>132</td>
<td>ERR_TX_ROLLBACK_RESOURCE</td>
<td>Transaction rolled back due to unavailable resource</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>133</td>
<td>ERR_TX_ROLLBACK_DEADLOCK</td>
<td>Transaction rolled back by detected deadlock</td>
</tr>
<tr>
<td>134</td>
<td>FATAL_REC_CHKPT_FILE_FAIL</td>
<td>Failure in accessing checkpoint file</td>
</tr>
<tr>
<td>135</td>
<td>FATAL_REC_ANCHOR_FILE_FAIL</td>
<td>Failure in accessing anchor file</td>
</tr>
<tr>
<td>136</td>
<td>FATAL_REC_LOG_FILE_FAIL</td>
<td>Failure in accessing log file</td>
</tr>
<tr>
<td>137</td>
<td>FATAL_REC_ARCHIVE_FILE_FAIL</td>
<td>Failure in accessing archive file</td>
</tr>
<tr>
<td>138</td>
<td>ERR_TX_SERIALIZEATION</td>
<td>Transaction serialization failure</td>
</tr>
<tr>
<td>139</td>
<td>ERR_TX_ROLLBACK_QUERY_CANCEL</td>
<td>Current operation cancelled by request and transaction rolled back</td>
</tr>
<tr>
<td>140</td>
<td>ERR_TX_INV_TID</td>
<td>Invalid write-transaction identifier</td>
</tr>
<tr>
<td>141</td>
<td>ERR_INV_LOG_FILE_FAIL</td>
<td>Failure in accessing invisible log file</td>
</tr>
<tr>
<td>142</td>
<td>ERR_TX_EXCEED_MAX_TX_NUM</td>
<td>Exceed max num of concurrent transactions</td>
</tr>
<tr>
<td>143</td>
<td>ERR_TX_SERIALIZEATION_WITH_TIMEOUT</td>
<td>Transaction serialization failure until timeout expires</td>
</tr>
<tr>
<td>144</td>
<td>ERR_TX_ROLLBACK_UNIQUE_VIOLATED</td>
<td>Transaction rollback</td>
</tr>
<tr>
<td>145</td>
<td>ERR_TX_DIST_FAILURE</td>
<td>Transaction distribution work failure</td>
</tr>
<tr>
<td>146</td>
<td>ERR_TX_LOCK_ACQUISITION_FAIL</td>
<td>Resource busy and NOWAIT specified</td>
</tr>
<tr>
<td>147</td>
<td>ERR_TX_DATA_LOG_INCONSISTENT</td>
<td>Inconsistency between data and log</td>
</tr>
<tr>
<td>148</td>
<td>ERR_TX_START_BLOCKED</td>
<td>Transaction start is blocked until Master_Restart finishes</td>
</tr>
<tr>
<td>149</td>
<td>ERR_TX_DIST_2PC_FAILURE</td>
<td>Distributed transaction commit failure</td>
</tr>
<tr>
<td>150</td>
<td>ERR_TX_SNAPSHOT_TOO_OLD</td>
<td>Statement cancelled or snapshot time-stamp already invalidated</td>
</tr>
<tr>
<td>151</td>
<td>ERR_TX_ROLLBACK_ROW_STORE_FULL</td>
<td>Transaction rollback</td>
</tr>
<tr>
<td>152</td>
<td>ERR_TX_ROLLBACK_ROW_STORE_FULL_DBAREQ</td>
<td>Transaction rollback</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>153</td>
<td>ERR_TX.Serialization_UNLOCK_REQUIRED</td>
<td>Transaction serialization failure</td>
</tr>
<tr>
<td>154</td>
<td>ERR_TX_INDEX_HANDLE_ACQUISITION_FAIL</td>
<td>Failure in acquiring index handle</td>
</tr>
<tr>
<td>155</td>
<td>ERR_TX_ROLLBACK_DEFERRED_FK</td>
<td>Transaction rollback</td>
</tr>
<tr>
<td>208</td>
<td>ERR_TX_XAER</td>
<td>XA transaction error</td>
</tr>
<tr>
<td>209</td>
<td>ERR_TX_XAER_ASYNC</td>
<td>Asynchronous operation error</td>
</tr>
<tr>
<td>210</td>
<td>ERR_TX_XAER_DUPLICATE</td>
<td>Duplicate XID</td>
</tr>
<tr>
<td>211</td>
<td>ERR_TX_XAER_INValid</td>
<td>Invalid argument of XA call</td>
</tr>
<tr>
<td>212</td>
<td>ERR_TX_XAER_NOTA</td>
<td>No valid XID</td>
</tr>
<tr>
<td>213</td>
<td>ERR_TX_XAER_OUTSIDE</td>
<td>Outside of global transaction</td>
</tr>
<tr>
<td>214</td>
<td>ERR_TX_XAERPROTO</td>
<td>Improper XA call sequence</td>
</tr>
<tr>
<td>215</td>
<td>ERR_TX_XAER_RMERROR</td>
<td>Resource manager error</td>
</tr>
<tr>
<td>216</td>
<td>ERR_TX_XAER_RMFAIL</td>
<td>Resource manager unavailable</td>
</tr>
<tr>
<td>217</td>
<td>ERR_TX_XA_RBPROTO</td>
<td>A protocol error occurred in the resource manager</td>
</tr>
<tr>
<td>218</td>
<td>ERR_TX_XA_RBROLLBACK</td>
<td>The rollback was caused by an unspecified reason</td>
</tr>
<tr>
<td>256</td>
<td>ERR_SQL</td>
<td>Sql processing error</td>
</tr>
<tr>
<td>257</td>
<td>ERR_SQL_PARSE</td>
<td>Sql syntax error</td>
</tr>
<tr>
<td>258</td>
<td>ERR_SQL_INSUFF_PRIV</td>
<td>Insufficient privilege</td>
</tr>
<tr>
<td>259</td>
<td>ERR_SQL_INV_TABLE</td>
<td>Invalid table name</td>
</tr>
<tr>
<td>260</td>
<td>ERR_SQL_INV_COLUMN</td>
<td>Invalid column name</td>
</tr>
<tr>
<td>261</td>
<td>ERR_SQL_INV_INDEX</td>
<td>Invalid index name</td>
</tr>
<tr>
<td>262</td>
<td>ERR_SQL_INV_QUERY</td>
<td>Invalid query name</td>
</tr>
<tr>
<td>263</td>
<td>ERR_SQL_INV_ALIAS</td>
<td>Invalid alias name</td>
</tr>
<tr>
<td>264</td>
<td>ERR_SQL_INV_DATATYPE</td>
<td>Invalid datatype</td>
</tr>
<tr>
<td>265</td>
<td>ERR_SQL_MISSING_EXP</td>
<td>Expression missing</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>266</td>
<td>ERR_SQL_INCNST_DATATYPE</td>
<td>Inconsistent datatype</td>
</tr>
<tr>
<td>267</td>
<td>ERR_SQL_LONG_LEN_TYPE</td>
<td>Specified length too long for its datatype</td>
</tr>
<tr>
<td>268</td>
<td>ERR_SQL_AMBG_COLUMN</td>
<td>Column ambiguously defined</td>
</tr>
<tr>
<td>269</td>
<td>ERR_SQL_MANY_VALUES</td>
<td>Too many values</td>
</tr>
<tr>
<td>270</td>
<td>ERR_SQL_FEW_VALUES</td>
<td>Not enough values</td>
</tr>
<tr>
<td>271</td>
<td>ERR_SQL_DPLC_ALIAS</td>
<td>Duplicate alias</td>
</tr>
<tr>
<td>272</td>
<td>ERR_SQL_DPLC_COLUMN</td>
<td>Duplicate column name</td>
</tr>
<tr>
<td>273</td>
<td>ERR_SQL_LONG_CHAR</td>
<td>Not a single character string</td>
</tr>
<tr>
<td>274</td>
<td>ERR_SQL_INS_LARGE_VALUE</td>
<td>Inserted value too large for column</td>
</tr>
<tr>
<td>275</td>
<td>ERR_SQL_NOT_FUNCTION</td>
<td>Aggregate function not allowed</td>
</tr>
<tr>
<td>276</td>
<td>ERR_SQL_NOT_SINGLE_GROUP</td>
<td>Missing aggregation or grouping</td>
</tr>
<tr>
<td>277</td>
<td>ERR_SQL_NOT_GROUP_EXP</td>
<td>Not a GROUP BY expression</td>
</tr>
<tr>
<td>278</td>
<td>ERR_SQL_NESTED_WO_GROUP</td>
<td>Nested group function without GROUP BY</td>
</tr>
<tr>
<td>279</td>
<td>ERR_SQL_TOO_DEEP_NESTED</td>
<td>Group function is nested</td>
</tr>
<tr>
<td>280</td>
<td>ERR_SQL_ORDER_EXCEED_NUM</td>
<td>ORDER BY item must be the number of a SELECT-list</td>
</tr>
<tr>
<td>281</td>
<td>ERR_SQL_OUTER_IN_OR</td>
<td>Outer join not allowed in operand of OR or IN</td>
</tr>
<tr>
<td>282</td>
<td>ERR_SQL_OUTER_CROSS_JOIN</td>
<td>Two tables cannot be outer-joined to each other</td>
</tr>
<tr>
<td>283</td>
<td>ERR_SQL_OUTER_MORE_TWO</td>
<td>A table may be outer joined to at most one other table</td>
</tr>
<tr>
<td>284</td>
<td>ERR_SQL_JOIN_NOT_MATCH</td>
<td>Join field does not match</td>
</tr>
<tr>
<td>285</td>
<td>ERR_SQL_INV_JOIN_PRED</td>
<td>Invalid join condition</td>
</tr>
<tr>
<td>286</td>
<td>ERR_SQL_LONG_IDENTIFIER</td>
<td>Identifier is too long</td>
</tr>
<tr>
<td>287</td>
<td>ERR_SQL_NOT_NULL</td>
<td>Cannot insert NULL or update to NULL</td>
</tr>
<tr>
<td>288</td>
<td>ERR_SQL_EXST_TABLE</td>
<td>Cannot use duplicate table name</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>289</td>
<td>ERR_SQL_EXST_INDEX</td>
<td>Cannot use duplicate index name</td>
</tr>
<tr>
<td>290</td>
<td>ERR_SQL_EXST_QUERY</td>
<td>Cannot use duplicate query name</td>
</tr>
<tr>
<td>291</td>
<td>ERR_SQL_NOT_POS_ARGUMENT</td>
<td>Argument identifier must be positive</td>
</tr>
<tr>
<td>292</td>
<td>ERR_SQL_FEW_ARGUMENT</td>
<td>Wrong number of arguments</td>
</tr>
<tr>
<td>293</td>
<td>ERR_SQL_INV_ARGUMENT</td>
<td>Argument type mismatch</td>
</tr>
<tr>
<td>294</td>
<td>ERR_SQL_MANY_PRIMARY_KEY</td>
<td>Cannot have more than one primary key</td>
</tr>
<tr>
<td>295</td>
<td>ERR_SQL_LONG_MULTIKEY</td>
<td>Too long multi key length</td>
</tr>
<tr>
<td>296</td>
<td>ERR_SQL_REP_TABLE_KEY</td>
<td>Replicated table must have a primary key</td>
</tr>
<tr>
<td>297</td>
<td>ERR_SQL_REP_UPDATE_KEY</td>
<td>Cannot update primary key field in replicated table</td>
</tr>
<tr>
<td>298</td>
<td>ERR_SQL_NOTDDL_STORE</td>
<td>Cannot store DDL</td>
</tr>
<tr>
<td>299</td>
<td>ERR_SQL_NOT_DROP_SYSIDX</td>
<td>Cannot drop index used for enforcement of unique/primary key</td>
</tr>
<tr>
<td>300</td>
<td>ERR_SQL_ARG_OUT_OF_RANGE</td>
<td>Argument index is out of range</td>
</tr>
<tr>
<td>301</td>
<td>ERR_SQL_UNIQUE_VIOLATED</td>
<td>Unique constraint violated</td>
</tr>
<tr>
<td>302</td>
<td>ERR_SQL_INV_CHAR_VAL</td>
<td>Invalid CHAR or VARCHAR value</td>
</tr>
<tr>
<td>303</td>
<td>ERR_SQL_INV_DATETIME_VAL</td>
<td>Invalid DATE</td>
</tr>
<tr>
<td>304</td>
<td>ERR_SQL_DIV_BY_ZERO</td>
<td>Division by zero undefined</td>
</tr>
<tr>
<td>305</td>
<td>ERR_SQL_SINGLE_ROW</td>
<td>Single-row query returns more than one row</td>
</tr>
<tr>
<td>306</td>
<td>ERR_SQL_INV_CURSOR</td>
<td>Invalid cursor</td>
</tr>
<tr>
<td>307</td>
<td>ERR_SQL_NUM_OUT_OF_RANGE</td>
<td>Numeric value out of range</td>
</tr>
<tr>
<td>308</td>
<td>ERR_SQL_EXST_COLUMN</td>
<td>Column name already exists</td>
</tr>
<tr>
<td>309</td>
<td>ERR_SQL_SUBQ_TOP_ORDERBY</td>
<td>Correlated subquery cannot have TOP or ORDER BY</td>
</tr>
<tr>
<td>310</td>
<td>ERR_SQL_IN_PROC</td>
<td>Sql error in procedure</td>
</tr>
<tr>
<td>311</td>
<td>ERR_SQL_DROP_ALL_COLUMNS</td>
<td>Cannot drop all columns in a table</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>312</td>
<td>ERR_SQL_SEQ_EXHAUST</td>
<td>Sequence is exhausted</td>
</tr>
<tr>
<td>313</td>
<td>ERR_SQL_INV_SEQ</td>
<td>Invalid sequence</td>
</tr>
<tr>
<td>314</td>
<td>ERR_SQL_OVERFLOW_NUMERIC</td>
<td>Numeric overflow</td>
</tr>
<tr>
<td>315</td>
<td>ERR_SQL_INV_SYNONYM</td>
<td>Invalid synonym</td>
</tr>
<tr>
<td>316</td>
<td>ERR_SQL_INV_NUM_ARG_FUNC</td>
<td>Wrong number of arguments in function invocation</td>
</tr>
<tr>
<td>317</td>
<td>ERR_SQL_NOT_MATCH_PLAN_TABLE</td>
<td>&quot;&quot;&quot;P_QUERYPLANS&quot;&quot; not exists nor valid format&quot;&quot;</td>
</tr>
<tr>
<td>318</td>
<td>ERR_SQL_DECIMAL_PRECISION</td>
<td>Decimal precision specifier is out of range</td>
</tr>
<tr>
<td>319</td>
<td>ERR_SQL_DECIMAL_SCALE</td>
<td>Decimal scale specifier is out of range</td>
</tr>
<tr>
<td>320</td>
<td>ERR_SQL_LOB_INDEX</td>
<td>Cannot create index on expression with datatype LOB</td>
</tr>
<tr>
<td>321</td>
<td>ERR_SQL_INV_VIEW</td>
<td>Invalid view name</td>
</tr>
<tr>
<td>322</td>
<td>ERR_SQL_EXST_VIEW</td>
<td>Cannot use duplicate view name</td>
</tr>
<tr>
<td>323</td>
<td>ERR_SQL_REP_DPLC_ID</td>
<td>Duplicate replication id</td>
</tr>
<tr>
<td>324</td>
<td>ERR_SQL_EXST_SEQ</td>
<td>Cannot use duplicate sequence name</td>
</tr>
<tr>
<td>325</td>
<td>ERR_SQL_ESC_SEQ</td>
<td>Invalid escape sequence</td>
</tr>
<tr>
<td>326</td>
<td>ERR_SQL_SEQ_CURRVAL</td>
<td>CURRVAL of given sequence is not yet defined in this session</td>
</tr>
<tr>
<td>327</td>
<td>ERR_SQL_CANNOT_EXPLAIN</td>
<td>Cannot explain plan of given statement</td>
</tr>
<tr>
<td>328</td>
<td>ERR_SQL_INV_FUNC_PROC</td>
<td>Invalid name of function or procedure</td>
</tr>
<tr>
<td>329</td>
<td>ERR_SQL_EXST_FUNC_PROC</td>
<td>Cannot use duplicate name of function or procedure</td>
</tr>
<tr>
<td>330</td>
<td>ERR_SQL_EXST_SYNONYM</td>
<td>Cannot use duplicate synonym name</td>
</tr>
<tr>
<td>331</td>
<td>ERR_SQL_EXST_USER</td>
<td>User name already exists</td>
</tr>
<tr>
<td>332</td>
<td>ERR_SQL_INV_USER</td>
<td>Invalid user name</td>
</tr>
<tr>
<td>333</td>
<td>ERR_SQL_COLUMN_NOT_ALLOWED_HERE</td>
<td>Column not allowed</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>334</td>
<td>ERR_SQL_INV_PRIV</td>
<td>Invalid user privilege</td>
</tr>
<tr>
<td>335</td>
<td>ERR_SQL_EXST_ALIAS</td>
<td>Field alias name already exists</td>
</tr>
<tr>
<td>336</td>
<td>ERR_SQL_INV_DEFAULT</td>
<td>Invalid default value</td>
</tr>
<tr>
<td>337</td>
<td>ERR_SQL INTO NOT ALLOWED</td>
<td>INTO clause not allowed for this SELECT statement</td>
</tr>
<tr>
<td>338</td>
<td>ERR_SQL_ZERO_LEN NOT ALLOWED</td>
<td>Zero-length columns are not allowed</td>
</tr>
<tr>
<td>339</td>
<td>ERR_SQL_INV_NUMBER</td>
<td>Invalid number</td>
</tr>
<tr>
<td>340</td>
<td>ERR_SQL_VAR_NOT_BOUND</td>
<td>Not all variables bound</td>
</tr>
<tr>
<td>341</td>
<td>ERR_SQLUNDERFLOW_NUMERIC</td>
<td>Numeric underflow</td>
</tr>
<tr>
<td>342</td>
<td>ERR_SQL_COLLATE_CONFLICT</td>
<td>Collation conflict</td>
</tr>
<tr>
<td>343</td>
<td>ERR_SQL_INV_COLLATE_NAME</td>
<td>Invalid collate name</td>
</tr>
<tr>
<td>344</td>
<td>ERR_SQL_LOADER_PARSE</td>
<td>Parse error in data loader</td>
</tr>
<tr>
<td>345</td>
<td>ERR_SQL NOT REP_TABLE</td>
<td>Not a replication table</td>
</tr>
<tr>
<td>346</td>
<td>ERR_SQL_INVREP_ID</td>
<td>Invalid replication id</td>
</tr>
<tr>
<td>347</td>
<td>ERR_SQL_INV_OPTION</td>
<td>Invalid option</td>
</tr>
<tr>
<td>348</td>
<td>ERR_SQL_INV_DATETIME_FORMAT</td>
<td>Invalid datetime format</td>
</tr>
<tr>
<td>349</td>
<td>ERR_SQL_CREATE_UNIQUE_INDEX</td>
<td>&quot;Cannot CREATE UNIQUE INDEX; duplicate key found&quot;</td>
</tr>
<tr>
<td>350</td>
<td>ERR_SQL DROP COL PRIMARY KEY</td>
<td>Cannot drop columns in the primary-key column list</td>
</tr>
<tr>
<td>351</td>
<td>ERR_SQL DROP MULTI COL UNIQUE</td>
<td>Column is referenced in a multi-column constraint</td>
</tr>
<tr>
<td>352</td>
<td>ERR_SQL CREATE UNIQUE INDEX ON CDX_TAB</td>
<td>Cannot create unique index on cdx table</td>
</tr>
<tr>
<td>353</td>
<td>ERR_SQL EXST UPDATE LOG GROUP</td>
<td>Update log group name already exists</td>
</tr>
<tr>
<td>354</td>
<td>ERR_SQL_INV UPDATE LOG GROUP NAME</td>
<td>Invalid update log group name</td>
</tr>
<tr>
<td>355</td>
<td>ERR_SQL UPDATE LOG TABLE KEY</td>
<td>The base table of the update log table must have a primary key</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>356</td>
<td>ERR_SQL_MAX_UPDATE_LOG_GROUP</td>
<td>Exceed maximum number of update log group</td>
</tr>
<tr>
<td>357</td>
<td>ERR_SQL_BASE_TABLE_ALREADY_HAS_ULT</td>
<td>The base table already has a update log table</td>
</tr>
<tr>
<td>358</td>
<td>ERR_SQL_ULT_CAN_NOT_HAVE_ULT</td>
<td>Update log table can not have a update log table</td>
</tr>
<tr>
<td>359</td>
<td>ERR_SQL_STR_LENGTH_TOO_LARGE</td>
<td>String is too long</td>
</tr>
<tr>
<td>360</td>
<td>ERR_SQL_VIEW_CHECK_VIOLATION</td>
<td>View WITH CHECK OPTION where-clause violation</td>
</tr>
<tr>
<td>361</td>
<td>ERR_SQL_VIEW_UPDATE_VIOLATION</td>
<td>Data manipulation operation not legal on this view</td>
</tr>
<tr>
<td>362</td>
<td>ERR_SQL_INV_SCHEMA</td>
<td>Invalid schema name</td>
</tr>
<tr>
<td>363</td>
<td>ERR_SQL_MAX_NUM_INDEX_COLUMN</td>
<td>Number of index columns exceeds its maximum</td>
</tr>
<tr>
<td>364</td>
<td>ERR_SQL_INV_PARTIAL_KEY_SIZE</td>
<td>Invalid partial key size</td>
</tr>
<tr>
<td>365</td>
<td>ERR_SQL_NO_MATCHING_UNIQUE_OR_PRIMARY_KEY</td>
<td>No matching primary key for this column list</td>
</tr>
<tr>
<td>366</td>
<td>ERR_SQL_NO_PRIMARY_KEY</td>
<td>Referenced table does not have a primary key</td>
</tr>
<tr>
<td>367</td>
<td>ERR_SQL_MISMATCH_OF_COLUMN_NUMBERS</td>
<td>Number of referencing columns must match referenced columns</td>
</tr>
<tr>
<td>368</td>
<td>ERR_SQL_TEMP_TABLE_WITH_UNIQUE</td>
<td>Unique constraint not allowed on temporary table</td>
</tr>
<tr>
<td>369</td>
<td>ERR_SQL_MAX_VIEW_DEPTH</td>
<td>Exceed maximum view depth limit</td>
</tr>
<tr>
<td>370</td>
<td>ERR_SQL_DIRECT_INSERT_WITH_UNIQUE_INDEX</td>
<td>Cannot perform DIRECT INSERT operation on table with unique indexes</td>
</tr>
<tr>
<td>371</td>
<td>ERR_SQL_XMLPARSE</td>
<td>Invalid XML document</td>
</tr>
<tr>
<td>372</td>
<td>ERR_SQL_XPATHPARSE</td>
<td>Invalid XPATH</td>
</tr>
<tr>
<td>373</td>
<td>ERR_SQL_INV_XML_DURATION</td>
<td>Invalid XML duration value</td>
</tr>
<tr>
<td>374</td>
<td>ERR_SQL_INV_XMLFUNCTION</td>
<td>Invalid XML function usage</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>375</td>
<td>ERR_SQL_INV_XML_INDEX_OPERATION</td>
<td>Invalid XML index operation</td>
</tr>
<tr>
<td>376</td>
<td>ERR.Sql_PYTHON</td>
<td>Python buildin procedure error</td>
</tr>
<tr>
<td>377</td>
<td>ERR_SQL_JIT</td>
<td>JIT operation error</td>
</tr>
<tr>
<td>378</td>
<td>ERR_SQL_INV_COLUMN_VIEW</td>
<td>Invalid column view</td>
</tr>
<tr>
<td>379</td>
<td>ERR_SQL_TABLE_SCHEMA_MISMATCH</td>
<td>Table schema mismatch</td>
</tr>
<tr>
<td>380</td>
<td>ERR_SQL_RUN_LEVEL_CHANGE</td>
<td>Fail to change run level</td>
</tr>
<tr>
<td>381</td>
<td>ERR_SQL_RESTART</td>
<td>Fail to restart</td>
</tr>
<tr>
<td>382</td>
<td>ERR_SQL_COLLECT_ALL_VERSIONS</td>
<td>Fail to collect all version garbage</td>
</tr>
<tr>
<td>383</td>
<td>ERR_SQL_INV_IDENTIFIER</td>
<td>Invalid identifier</td>
</tr>
<tr>
<td>384</td>
<td>ERR_SQL_TOO_LONG_CONSTANT</td>
<td>String is too long</td>
</tr>
<tr>
<td>385</td>
<td>ERR_SQL_RESTORE_SESSION</td>
<td>Could not restore session</td>
</tr>
<tr>
<td>386</td>
<td>ERR_SQL_EXST_SCHEMA</td>
<td>Cannot use duplicate schema name</td>
</tr>
<tr>
<td>387</td>
<td>ERR_SQL_AMBIG_TABLE</td>
<td>Table ambiguously defined</td>
</tr>
<tr>
<td>388</td>
<td>ERR_SQL_EXST_ROLE</td>
<td>Role already exists</td>
</tr>
<tr>
<td>389</td>
<td>ERR_SQL_INV_ROLE</td>
<td>Invalid role name</td>
</tr>
<tr>
<td>390</td>
<td>ERR_SQL_INV_USERTYPE</td>
<td>Invalid user type</td>
</tr>
<tr>
<td>391</td>
<td>ERR_SQL_INV_USABLE_VIEW</td>
<td>Invalidated view</td>
</tr>
<tr>
<td>392</td>
<td>ERR_SQL_CYCLIC_ROLES</td>
<td>Can't assign cyclic role</td>
</tr>
<tr>
<td>393</td>
<td>ERR_SQL_NO_GRANT_OPTION_FOR_ROLE</td>
<td>Roles must not receive a privilege with grant option</td>
</tr>
<tr>
<td>394</td>
<td>ERR_SQL_CANT_REVOKE_ROLE</td>
<td>Error revoking role</td>
</tr>
<tr>
<td>395</td>
<td>ERR_SQL_INV_USER_DEFINED_TYPE</td>
<td>Invalid user-defined type name</td>
</tr>
<tr>
<td>396</td>
<td>ERR_SQL_EXST_USER_DEFINED_TYPE</td>
<td>Cannot use duplicate user-defined type name</td>
</tr>
<tr>
<td>397</td>
<td>ERR_SQL_INV_OBJ_NAME</td>
<td>Invalid object name</td>
</tr>
<tr>
<td>398</td>
<td>ERR_SQL_MANY_ORDER_BY</td>
<td>Cannot have more than one order by</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>399</td>
<td>ERR_SQL_TOO_DEEP_ROLE_TREE</td>
<td>Role tree too deep</td>
</tr>
<tr>
<td>400</td>
<td>ERR_SQL_INSERT_ONLY_TABLE_WITH_PRIMARY_KEY</td>
<td>Primary key not allowed on insert-only table</td>
</tr>
<tr>
<td>401</td>
<td>ERR_SQL_INSERT_ONLY_TABLE_WITH_UNIQUE</td>
<td>Unique constraint not allowed on insert-only table</td>
</tr>
<tr>
<td>402</td>
<td>ERR_SQL_DROPPED_USER</td>
<td>The user was already dropped before query execution</td>
</tr>
<tr>
<td>403</td>
<td>ERR_SQL_INTERNAL_ERROR</td>
<td>Internal error</td>
</tr>
<tr>
<td>404</td>
<td>ERR_SQL_INV_STRUCTURED_PRIVILEGE_NAME</td>
<td>Invalid (non-existent) structured privilege name</td>
</tr>
<tr>
<td>405</td>
<td>ERR_SQL_DUP_STRUCTURED_PRIVILEGE_NAME</td>
<td>Cannot use duplicate structured privilege name</td>
</tr>
<tr>
<td>406</td>
<td>ERR_SQL_CANT_UPDATE_GEN_COL</td>
<td>INSERT</td>
</tr>
<tr>
<td>407</td>
<td>ERR_SQL_INV_DATE_FORMAT</td>
<td>Invalid date format</td>
</tr>
<tr>
<td>408</td>
<td>ERR_SQL_PASS_OR_PARAMETER_NEEDED</td>
<td>Password or parameter required for user</td>
</tr>
<tr>
<td>409</td>
<td>ERR_SQL_TOO_MANY_PARAMETER_VALUES</td>
<td>Multiple values for a parameter not supported</td>
</tr>
<tr>
<td>410</td>
<td>ERR_SQL_INV_PRIVILEGE_NAME_SPACE</td>
<td>Invalid privilege namespace</td>
</tr>
<tr>
<td>411</td>
<td>ERR_SQL_INV_TABLE_TYPE</td>
<td>Invalid table type</td>
</tr>
<tr>
<td>412</td>
<td>ERR_SQL_INV_PASSWORD_LAYOUT</td>
<td>Invalid password layout</td>
</tr>
<tr>
<td>413</td>
<td>ERR_SQL_PASSWORD_REUSED</td>
<td>Last n passwords can not be reused</td>
</tr>
<tr>
<td>414</td>
<td>ERR_SQL_ALTER_PASSWORD_NEEDED</td>
<td>User is forced to change password</td>
</tr>
<tr>
<td>415</td>
<td>ERR_SQL_USER_DEACTIVATED</td>
<td>User is deactivated</td>
</tr>
<tr>
<td>416</td>
<td>ERR_SQL_USER_LOCKED</td>
<td>&quot;User is locked; try again later&quot;</td>
</tr>
<tr>
<td>417</td>
<td>ERR_SQL_CANT_DROP_WITHOUT.CASCADE</td>
<td>Can't drop without CASCADE specification</td>
</tr>
<tr>
<td>418</td>
<td>ERR_SQL_INV_VIEW_QUERY</td>
<td>Invalid view query for creation</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>419</td>
<td>ERR_SQL_CANT_DROP_WITH_RESTRICT</td>
<td>Can't drop with RESTRICT specification</td>
</tr>
<tr>
<td>420</td>
<td>ERR_SQL_ALTER_PASSWORD_NOT_ALLOWED</td>
<td>Password change currently not allowed</td>
</tr>
<tr>
<td>421</td>
<td>ERR_SQL_FULLTEXT_INDEX</td>
<td>Cannot create fulltext index</td>
</tr>
<tr>
<td>422</td>
<td>ERR_SQL_MIXED_PRIVILEGE_NAMESPACES</td>
<td>Privileges must be either all SQL or all from one namespace</td>
</tr>
<tr>
<td>423</td>
<td>ERR_SQL_LVC</td>
<td>AFL error</td>
</tr>
<tr>
<td>424</td>
<td>ERR_SQL_INV_PACKAGE</td>
<td>Invalid name of package</td>
</tr>
<tr>
<td>425</td>
<td>ERR_SQL_EXST_PACKAGE</td>
<td>Duplicate package name</td>
</tr>
<tr>
<td>426</td>
<td>ERR_SQL_NUM_COLUMN_MISMATCH</td>
<td>Number of columns mismatch</td>
</tr>
<tr>
<td>427</td>
<td>ERR_SQL_CANT_RESERVE_INDEX_ID</td>
<td>Can not reserve index id any more</td>
</tr>
<tr>
<td>428</td>
<td>ERR_INV_QUERY_PLAN_ID</td>
<td>Invalid query plan id</td>
</tr>
<tr>
<td>429</td>
<td>ERR_SQL_INTEGRITY_CHECK_FAILED</td>
<td>Integrity check failed</td>
</tr>
<tr>
<td>430</td>
<td>ERR_SQL_INV_USABLE_PROC</td>
<td>Invalidated procedure</td>
</tr>
<tr>
<td>431</td>
<td>WRN_SQL_NEARLY_EXPIRED_PASSWORD</td>
<td>User's password will expire within few days</td>
</tr>
<tr>
<td>432</td>
<td>WRN_SQL_DEPRECATED_SYNTAX</td>
<td>This syntax has been deprecated and will be removed in next release</td>
</tr>
<tr>
<td>433</td>
<td>ERR_SQL_NOT_NULL_CONSTRAINT</td>
<td>Null value found</td>
</tr>
<tr>
<td>434</td>
<td>ERR_SQL_INV_OBJECT</td>
<td>Invalid object ID</td>
</tr>
<tr>
<td>435</td>
<td>ERR_SQL_INV_EXP</td>
<td>Invalid expression</td>
</tr>
<tr>
<td>436</td>
<td>ERR_SQL_SET_SYSTEM_LICENSE</td>
<td>Could not set system license</td>
</tr>
<tr>
<td>437</td>
<td>ERR_SQL_ONLY_LICENSE_HANDLING</td>
<td>Only commands for license handling are allowed in current state</td>
</tr>
<tr>
<td>438</td>
<td>ERR_SQL_INVALID_USER_PARAMETER_VALUE</td>
<td>Invalid user parameter value</td>
</tr>
<tr>
<td>439</td>
<td>ERR_SQL_COMPOSITE_ERROR</td>
<td>Composite error</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>440</td>
<td>ERR_SQL_TABLE_TYPE_CONVERSION_ERROR</td>
<td>Table type conversion error</td>
</tr>
<tr>
<td>441</td>
<td>WRN_SQL_DEPRECATED_FEATURE</td>
<td>This feature has been deprecated and will be removed in next release</td>
</tr>
<tr>
<td>442</td>
<td>ERR_SQL_MAX_NUM_COLUMN</td>
<td>Number of columns exceeds its maximum</td>
</tr>
<tr>
<td>443</td>
<td>ERR_SQL_INV_CALC_SCENARIO</td>
<td>Invalid calculation scenario name</td>
</tr>
<tr>
<td>444</td>
<td>ERR_SQL_PACKMAN</td>
<td>Package manager error</td>
</tr>
<tr>
<td>445</td>
<td>ERR_SQL_INV_TRIGGER</td>
<td>Invalid trigger name</td>
</tr>
<tr>
<td>446</td>
<td>ERR_SQL_EXST_TRIGGER</td>
<td>Cannot use duplicate trigger name</td>
</tr>
<tr>
<td>447</td>
<td>ERR_SQL_BACKUP_FAILED</td>
<td>Backup could not be completed</td>
</tr>
<tr>
<td>448</td>
<td>ERR_SQL_RECOVERY_FAILED</td>
<td>Recovery could not be completed</td>
</tr>
<tr>
<td>449</td>
<td>ERR_SQL_RECOVERY_STRATEGY</td>
<td>Recovery strategy could not be determined</td>
</tr>
<tr>
<td>450</td>
<td>ERR_SQL_UNSET_SYSTEM_LICENSE</td>
<td>Failed to unset system license</td>
</tr>
<tr>
<td>451</td>
<td>ERR_SQL_NOT_ALLOWED_SUBJ_TAB_ACCESS_TRIGGER</td>
<td>Modification of subject table in trigger not allowed</td>
</tr>
<tr>
<td>452</td>
<td>ERR_SQL_INV_BACKUPID</td>
<td>Invalid backup id</td>
</tr>
<tr>
<td>453</td>
<td>ERR_SQL_USER_WITHOUT_PASSWORD</td>
<td>User does not have a password</td>
</tr>
<tr>
<td>454</td>
<td>WRN_SQL_WRONG_HINT_SYNTAX</td>
<td>Wrong hint syntax</td>
</tr>
<tr>
<td>455</td>
<td>ERR_SQL_READ_ONLY_SESSION_VARIABLE</td>
<td>The predefined session variable cannot be set via SET command</td>
</tr>
<tr>
<td>456</td>
<td>ERR_SQL_NOT_ALLOWED_FOR_SPECIAL_ROLE</td>
<td>Not allowed for this role</td>
</tr>
<tr>
<td>457</td>
<td>ERR_SQL_DPLC_CONSTRAINT</td>
<td>Duplicate constraint name</td>
</tr>
<tr>
<td>458</td>
<td>ERR_SQL_UNSUPPORTED_FUNCTION</td>
<td>Unsupported function included</td>
</tr>
<tr>
<td>459</td>
<td>ERR_SQL_INV_USABLE_FUNC</td>
<td>Invalidated function</td>
</tr>
<tr>
<td>460</td>
<td>ERR_SQL_INV_PRIVILEGE_FOR_OBJECT</td>
<td>Invalid privilege for object</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>461</td>
<td>ERR_SQL_FK_NOT_FOUND</td>
<td>Foreign key constraint violation</td>
</tr>
<tr>
<td>462</td>
<td>ERR_SQL_FK_ON_UPDATE_DELETE_FAILED</td>
<td>Failed on update or delete by foreign key constraint violation</td>
</tr>
<tr>
<td>463</td>
<td>ERR_SQL_MAX_NUM_TABLE</td>
<td>Number of tables exceeds its maximum</td>
</tr>
<tr>
<td>464</td>
<td>ERR_SQL_MAX_PARSE_TREE_DEPTH</td>
<td>SQL internal parse tree depth exceeds its maximum</td>
</tr>
<tr>
<td>465</td>
<td>ERR_SQL_INV_USABLE_TRIGGER</td>
<td>Cannot execute trigger</td>
</tr>
<tr>
<td>466</td>
<td>ERR_SQL_CREDENTIAL_NOT_FOUND</td>
<td>No credential found</td>
</tr>
<tr>
<td>467</td>
<td>ERR_SQL_PARAM_VARIABLE</td>
<td>Cannot use parameter variable</td>
</tr>
<tr>
<td>468</td>
<td>ERR_SQL_HINT</td>
<td>Hint error</td>
</tr>
<tr>
<td>469</td>
<td>ERR_SQL_INV_SRC_DATATYPE</td>
<td>Unsupported datatype on source</td>
</tr>
<tr>
<td>470</td>
<td>ERR_SQL_INV_DATA_SOURCE_CONF</td>
<td>Invalid data source configuration</td>
</tr>
<tr>
<td>471</td>
<td>ERR_SQL_INV_DATA_SOURCE</td>
<td>Invalid data source name</td>
</tr>
<tr>
<td>472</td>
<td>ERR_SQL_EXST_DATA_SOURCE</td>
<td>Cannot use duplicate data source name</td>
</tr>
<tr>
<td>473</td>
<td>ERR_SQL_ADAPTER_CONFIGURATION</td>
<td>Invalid adapter configuration</td>
</tr>
<tr>
<td>474</td>
<td>ERR_SQL_INV_ADAPTER</td>
<td>Invalid adapter name</td>
</tr>
<tr>
<td>475</td>
<td>ERR_SQL_EXST_ADAPTER</td>
<td>Cannot use duplicate adapter name</td>
</tr>
<tr>
<td>476</td>
<td>ERR_SQL_INV_REMOTE_OBJECT</td>
<td>Invalid remote object name</td>
</tr>
<tr>
<td>477</td>
<td>ERR_SQL_CREDENTIAL_EXISTS</td>
<td>Credential exists</td>
</tr>
<tr>
<td>478</td>
<td>ERR_SQL_UDF_RUNTIME</td>
<td>User defined function runtime error</td>
</tr>
<tr>
<td>479</td>
<td>ERR_SQL_INV_SPATIAL_ATTRIBUTE</td>
<td>Invalid spatial attribute</td>
</tr>
<tr>
<td>480</td>
<td>ERR_SQL_INV_SPATIAL_UNIT</td>
<td>Invalid spatial unit of measure name</td>
</tr>
<tr>
<td>481</td>
<td>ERR_SQL_EXST_SPATIAL_UNIT</td>
<td>Cannot use duplicate spatial unit of measure name</td>
</tr>
<tr>
<td>482</td>
<td>ERR_SQL_INV_SPATIAL_REF_SYS</td>
<td>Invalid spatial reference system name</td>
</tr>
<tr>
<td>483</td>
<td>ERR_SQL_EXST_SPATIAL_REF_SYS</td>
<td>Cannot use duplicate spatial reference system name</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>484</td>
<td>ERR_SQL_SESSION_GROUP_COMMAND_FAILURE</td>
<td>Invalid session group command</td>
</tr>
<tr>
<td>485</td>
<td>ERR_SQL_INV_STRUCTURED_PRIVILEGE_DEFINITION</td>
<td>Invalid definition of structured privilege</td>
</tr>
<tr>
<td>486</td>
<td>WRN_SQL_IMPORT_PARTIALLY_FAILED</td>
<td>Some rows have failed to be imported</td>
</tr>
<tr>
<td>487</td>
<td>ERR_SQL_IMPORT_PARTIALLY_FAILED</td>
<td>Some rows have failed to be imported</td>
</tr>
<tr>
<td>488</td>
<td>ERR_SQL_INV_DATABASE</td>
<td>Invalid database name</td>
</tr>
<tr>
<td>489</td>
<td>ERR_SQL_INV_EPMMODEL</td>
<td>Invalid EPM Model name</td>
</tr>
<tr>
<td>490</td>
<td>ERR_SQL_EXT_EPMMODEL</td>
<td>Cannot use duplicate EPM Model name</td>
</tr>
<tr>
<td>491</td>
<td>ERR_SQL_INV_EPMMODEL_DEF</td>
<td>Invalid EPM Model definition</td>
</tr>
<tr>
<td>492</td>
<td>ERR_SQL_INV_EPMQUERYSOURCE</td>
<td>Invalid EPM Query Source name</td>
</tr>
<tr>
<td>493</td>
<td>ERR_SQL_EXT_EPMQUERYSOURCE</td>
<td>Cannot use duplicate EPM Query Source name</td>
</tr>
<tr>
<td>494</td>
<td>ERR_SQL_INV_EPMQUERYSOURCE_DEF</td>
<td>Invalid EPM Query Source definition</td>
</tr>
<tr>
<td>495</td>
<td>ERROR_SQL_INVALID_CONV_TO_EXTENDED</td>
<td>Table already Extended with right delta option</td>
</tr>
<tr>
<td>496</td>
<td>ERROR_SQL_INVALID_CONV_FROM_EXTENDED</td>
<td>Table already Non-Extended</td>
</tr>
<tr>
<td>497</td>
<td>ERROR_EXTENDED_STORAGE_NOT_FOUND</td>
<td>Extended Storage not found for table:</td>
</tr>
<tr>
<td>498</td>
<td>ERR_SQL_IMPORT_FAIL_ON_MAX_RECORD_SIZE_CHECK</td>
<td>Memory for a record exceeds the limit</td>
</tr>
<tr>
<td>499</td>
<td>ERR_SQL_INV_C2C</td>
<td>Invalid stacked column search</td>
</tr>
<tr>
<td>500</td>
<td>ERR_SQL_REQUIRE_PREDICATE</td>
<td>Predicates are required in a where clause</td>
</tr>
<tr>
<td>501</td>
<td>ERR_SQL_SERIES_INVALID_SPEC</td>
<td>Invalid series data specification:</td>
</tr>
<tr>
<td>502</td>
<td>ERR_SQL_INV_TASK</td>
<td>Invalid name of task</td>
</tr>
<tr>
<td>503</td>
<td>ERR_SQL_EXT_TASK</td>
<td>Cannot use duplicate name of task</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>504</td>
<td>ERR_SQL_INV_ADAPTER_LOCATION</td>
<td>Invalid adapter location</td>
</tr>
<tr>
<td>505</td>
<td>ERR_SQL_LAST_ADAPTER_LOCATION</td>
<td>Cannot remove last location of adapter</td>
</tr>
<tr>
<td>506</td>
<td>ERR_SQL_SYSTEM_ADAPTER</td>
<td>Invalid create</td>
</tr>
<tr>
<td>507</td>
<td>ERR_SQL_INV_AGENT</td>
<td>Invalid agent name</td>
</tr>
<tr>
<td>508</td>
<td>ERR_SQL_EXST_AGENT</td>
<td>Cannot use duplicate agent name</td>
</tr>
<tr>
<td>509</td>
<td>ERR_SQL_INV_AGENT_PROPS</td>
<td>Invalid agent properties</td>
</tr>
<tr>
<td>510</td>
<td>ERR_SQL_TEMP_TABLE_IN_USE</td>
<td>Cannot alter global temporary table in use or create/alter/drop index on the table</td>
</tr>
<tr>
<td>512</td>
<td>ERR_REP</td>
<td>Replication error</td>
</tr>
<tr>
<td>513</td>
<td>ERR_SQL_REP_ALREADY_ACTIVE</td>
<td>Cannot execute DDL statement on replication table while replicating</td>
</tr>
<tr>
<td>514</td>
<td>FATAL_REP_ANCHOR_FILE_FAIL</td>
<td>Failure in accessing anchor file</td>
</tr>
<tr>
<td>515</td>
<td>FATAL_REP_LOG_FILE_FAIL</td>
<td>Failure in accessing log file</td>
</tr>
<tr>
<td>516</td>
<td>ERR_REP_TABLE_HAVE_NOT_REPORT_TABLE</td>
<td>Replication table has not conflict report table</td>
</tr>
<tr>
<td>517</td>
<td>ERR_REPORT_TABLE_ALREADY_ENABLED</td>
<td>Conflict report table already enabled</td>
</tr>
<tr>
<td>518</td>
<td>ERR_REPORT_TABLE_ALREADY_DISABLED</td>
<td>Conflict report table already disabled</td>
</tr>
<tr>
<td>544</td>
<td>ERR_RS_PARTITION</td>
<td>Partition error</td>
</tr>
<tr>
<td>545</td>
<td>ERR_RS_PARTITION_INVALID_SPEC</td>
<td>Invalid partition spec</td>
</tr>
<tr>
<td>546</td>
<td>ERR_RS_PARTITION_INVALID_ID</td>
<td>Invalid partition id</td>
</tr>
<tr>
<td>547</td>
<td>ERR_RS_PARTITION_INVALID_PROPERTY</td>
<td>Invalid use of partition property</td>
</tr>
<tr>
<td>548</td>
<td>ERR_RS_PARTITION_INVALID_RANGE</td>
<td>Invalid partition range</td>
</tr>
<tr>
<td>549</td>
<td>ERR_RS_PARTITION_INVALID_TYPE</td>
<td>Invalid partition type</td>
</tr>
<tr>
<td>550</td>
<td>ERR_RS_PARTITION_INVALID_COLUMN</td>
<td>Invalid partition column</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>551</td>
<td>ERR_RS_PARTITION_INVALID_LEVEL</td>
<td>Invalid partition level</td>
</tr>
<tr>
<td>576</td>
<td>ERR_API</td>
<td>Api error</td>
</tr>
<tr>
<td>577</td>
<td>ERR_API_NO_CURSOR_FWD</td>
<td>Cursor type of forward is not allowed</td>
</tr>
<tr>
<td>578</td>
<td>ERR_API_INV_STATEMENT</td>
<td>Invalid statement</td>
</tr>
<tr>
<td>579</td>
<td>ERR_API_EXCEED_MAX_GROUP_SIZE</td>
<td>Exceed maximum batch size</td>
</tr>
<tr>
<td>580</td>
<td>ERR_API_VERSION_INCOMPATIBLE</td>
<td>Server rejected the connection(protocol version mismatch)</td>
</tr>
<tr>
<td>581</td>
<td>ERR_APIONLY_SINGLE_STATEMENT</td>
<td>This function can be called only in the case of single statement</td>
</tr>
<tr>
<td>582</td>
<td>ERR_API_INV_RETURN_OBJECT</td>
<td>This query does not have result set</td>
</tr>
<tr>
<td>583</td>
<td>ERR_API_CONNECTION_DOES_NOT_EXIST</td>
<td>Connection does not exist</td>
</tr>
<tr>
<td>584</td>
<td>ERR_API_NO_MORE_LOB_DATA</td>
<td>No more LOB data</td>
</tr>
<tr>
<td>585</td>
<td>ERR_API_OPERATION_NOT_PERMITTED</td>
<td>Operation is not permitted</td>
</tr>
<tr>
<td>586</td>
<td>ERR_API_INV_PARAMETER_FROM_SERVER</td>
<td>Invalid parameter is received from server</td>
</tr>
<tr>
<td>587</td>
<td>ERR_API_INV_RESULTSET</td>
<td>Result set is currently invalid</td>
</tr>
<tr>
<td>588</td>
<td>ERR_API_RESULTSET_NEXT_NOT_CALLED</td>
<td>Next() is not called for this result set</td>
</tr>
<tr>
<td>589</td>
<td>ERR_API_TOO_MANY_PARAMETERS</td>
<td>Too many parameters are set</td>
</tr>
<tr>
<td>590</td>
<td>ERR_API_MISSING_PARAMETER</td>
<td>Some parameters are missing</td>
</tr>
<tr>
<td>591</td>
<td>ERR_API_INTERNAL_ERROR</td>
<td>Internal error</td>
</tr>
<tr>
<td>592</td>
<td>ERR_API_NOT_SUPPORTED_TYPE_CONV</td>
<td>Not supported type conversion</td>
</tr>
<tr>
<td>593</td>
<td>ERR_API_REMOTE_ONLY</td>
<td>Remote-only function</td>
</tr>
<tr>
<td>594</td>
<td>ERR_API_NO_MORE_RESULT_ROW</td>
<td>No more result row in result set</td>
</tr>
<tr>
<td>595</td>
<td>ERR_API_NOT_OUT_PARAM</td>
<td>Specified parameter is not output parameter</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>596</td>
<td>ERR_API_LOB_STREAM_NOT_PERMITTED</td>
<td>LOB streaming is not permitted in auto-commit mode</td>
</tr>
<tr>
<td>597</td>
<td>ERR_API_SESSION_CONTEXT_ERROR</td>
<td>Session context error</td>
</tr>
<tr>
<td>598</td>
<td>ERR_API_EXTERNAL_EXECUTION_FAILURE</td>
<td>Failed to execute the external statement</td>
</tr>
<tr>
<td>599</td>
<td>ERR_API_NOT_INITIALIZED</td>
<td>Session layer is not initialized yet</td>
</tr>
<tr>
<td>600</td>
<td>ERR_API_CALL_ROUTING_FAILURE</td>
<td>Failed routed execution</td>
</tr>
<tr>
<td>601</td>
<td>ERR_API_TOO_MANY_SESSION_VARIABLES</td>
<td>Too many session variables are set</td>
</tr>
<tr>
<td>602</td>
<td>ERR_API_READONLY_SESSION_VARIABLE</td>
<td>Cannot set read-only session variable</td>
</tr>
<tr>
<td>603</td>
<td>ERR_API_INV_LOB</td>
<td>Invalid LOB</td>
</tr>
<tr>
<td>604</td>
<td>ERR_API_REMOTE_TEMP_TABLE</td>
<td>Remote temp table access failure</td>
</tr>
<tr>
<td>605</td>
<td>ERR_API_INV_XA_JOIN</td>
<td>Invalid xa join request</td>
</tr>
<tr>
<td>606</td>
<td>ERR_API_EXCEED_MAX_LOB_SIZE</td>
<td>Exceed maximum LOB size</td>
</tr>
<tr>
<td>607</td>
<td>ERR_API_CLEANUP_RESOURCE</td>
<td>Failed to cleanup resources</td>
</tr>
<tr>
<td>608</td>
<td>ERR_API_EXCEED_MAX_PREPARED_STATEMENT</td>
<td>Exceed maximum number of prepared statements</td>
</tr>
<tr>
<td>609</td>
<td>ERR_APIINVALID_CESU8_STRING</td>
<td>Invalid CESU-8 string</td>
</tr>
<tr>
<td>610</td>
<td>ERR_API_ROLLBACK_FAILURE</td>
<td>Transaction rollback failure</td>
</tr>
<tr>
<td>611</td>
<td>ERR_API_SESSION_VARIABLE_VALUE_LENGTH_EXCEEDED</td>
<td>Maximum length of value for session variable exceeded</td>
</tr>
<tr>
<td>612</td>
<td>ERR_API_SESSION_VARIABLE_KEY_LENGTH_EXCEEDED</td>
<td>Maximum length of key for session variable exceeded</td>
</tr>
<tr>
<td>613</td>
<td>ERR_API_TIMEOUT</td>
<td>Execution aborted by timeout</td>
</tr>
<tr>
<td>614</td>
<td>ERR_API_ACCESS_ENCRYPTED_DATA</td>
<td>Encrypted data access is not permitted</td>
</tr>
<tr>
<td>615</td>
<td>ERR_API_REMOTE_CONNECTION_DOES_NOT_EXIST</td>
<td>Remote connection does not exist</td>
</tr>
<tr>
<td>616</td>
<td>ERR_API_REJECTED_BY_WORKLOAD_CLASS</td>
<td>Rejected by workload class configuration</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>640</td>
<td>ERR_SQL_2</td>
<td>Sql processing error</td>
</tr>
<tr>
<td>641</td>
<td>ERR_SQL_INV_REMOTE_SUBSCRIPTION</td>
<td>Invalid remote subscription name</td>
</tr>
<tr>
<td>642</td>
<td>ERR_SQL_EXST_REMOTE_SUBSCRIPTION</td>
<td>Cannot use duplicate remote subscription name</td>
</tr>
<tr>
<td>643</td>
<td>ERR_SQL_INV_REMOTE_SUBSCRIPTION_DEF</td>
<td>Invalid remote subscription definition</td>
</tr>
<tr>
<td>644</td>
<td>ERR_SQL_EXST_REMOTE_SOURCE_ADAPTER_LOCATION</td>
<td>Remote source refers to the adapter location</td>
</tr>
<tr>
<td>645</td>
<td>ERR_SQL_EXST_REMOTE_SOURCE_ACTIVE_SUBSCRIPTIONS</td>
<td>Remote source has active remote subscriptions:</td>
</tr>
<tr>
<td>646</td>
<td>ERR_SQL_INV_USABLE_TASK</td>
<td>Invalidated task</td>
</tr>
<tr>
<td>647</td>
<td>ERR_SQL_NOT_ALLOWED_SYNTAX_FOR_TRIGGER</td>
<td>Not supported syntax in trigger</td>
</tr>
<tr>
<td>648</td>
<td>ERR_SQL_TRIGGER_AND_PROC_NESTING_DEPTH_EXCEEDED</td>
<td>Nesting depth of trigger and procedure is exceeded</td>
</tr>
<tr>
<td>649</td>
<td>ERR_SQL_QUERY_PINNED_PLAN</td>
<td>Pinned plan error</td>
</tr>
<tr>
<td>650</td>
<td>ERR_SQL_QUERY_REMOVE_PINNED_PLAN</td>
<td>Remove pinned plan error</td>
</tr>
<tr>
<td>651</td>
<td>ERR_SQL_EXST_OBJECT</td>
<td>Cannot use duplicate object name</td>
</tr>
<tr>
<td>652</td>
<td>ERR_SQL_AMBG_SCHEMA</td>
<td>Schema ambiguously defined</td>
</tr>
<tr>
<td>653</td>
<td>ERR_SQL_SET_ROW_ORDER</td>
<td>Row order already set on table</td>
</tr>
<tr>
<td>654</td>
<td>ERR_SQL_NO_ROW_ORDER</td>
<td>No row order on table set</td>
</tr>
<tr>
<td>655</td>
<td>ERR_SQL_LICENSING_RUNTIME</td>
<td>Licensing error</td>
</tr>
<tr>
<td>656</td>
<td>ERR_SQL_LONG_PROPERTY</td>
<td>Property value too long</td>
</tr>
<tr>
<td>657</td>
<td>ERR_SQL_CANCEL_TASK_TIMEOUT_REACHED</td>
<td>Request to cancel task was sent but task did not cancel before timeout was reached</td>
</tr>
<tr>
<td>658</td>
<td>ERR_SQL_CANNOT_MUTATE_TABLE_DURING_FK_EXECUTION</td>
<td>Cannot mutate the table during trigger or foreign key execution</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>659</td>
<td>ERR_SQL_EXST_WORKLOAD_CLASS</td>
<td>Cannot use duplicate workload class name</td>
</tr>
<tr>
<td>660</td>
<td>ERR_SQL_INV_WORKLOAD_CLASS</td>
<td>Invalid workload class name</td>
</tr>
<tr>
<td>661</td>
<td>ERR_SQL_EXST_WORKLOAD_MAPPING</td>
<td>Cannot use duplicate workload mapping name</td>
</tr>
<tr>
<td>662</td>
<td>ERR_SQL_INV_WORKLOAD_MAPPING</td>
<td>Invalid workload mapping name</td>
</tr>
<tr>
<td>663</td>
<td>ERR_SQL_CONNECT_NOT_ALLOWED</td>
<td>User not allowed to connect from client</td>
</tr>
<tr>
<td>664</td>
<td>ERR_SQL_INV_AGENT_GROUP</td>
<td>Invalid agent group name</td>
</tr>
<tr>
<td>665</td>
<td>ERR_SQL_EXST_AGENT_GROUP</td>
<td>Cannot use duplicate agent group name</td>
</tr>
<tr>
<td>666</td>
<td>ERR_SQL_AGENT_GROUP_NOT_EMPTY</td>
<td>Agents are still set to this agent group.</td>
</tr>
<tr>
<td>667</td>
<td>ERR_SQL_TEXT_MINING_FAILURE</td>
<td>Text mining error</td>
</tr>
<tr>
<td>668</td>
<td>ERR_SQL_2D_POINTS_SUPPORTED_ONLY</td>
<td>ST_Point columns support 2-dimensional points only</td>
</tr>
<tr>
<td>669</td>
<td>ERR_SQL_SPATIAL_ERROR</td>
<td>Spatial error</td>
</tr>
<tr>
<td>670</td>
<td>ERR_SQL_PART_NOT_EXIST</td>
<td>Part does not exist</td>
</tr>
<tr>
<td>671</td>
<td>ERR_SQL_EXST_LIBRARY</td>
<td>Cannot use duplicate library name</td>
</tr>
<tr>
<td>672</td>
<td>ERR_SQL_DPLC_ASSOCIATION</td>
<td>Duplicate association name</td>
</tr>
<tr>
<td>673</td>
<td>ERR_SQL_INV_GRAPH_WORKSPACE</td>
<td>Invalid graph workspace name</td>
</tr>
<tr>
<td>674</td>
<td>WRN_SQL_EXPORT_SKIP_CROSSDB_OBJECT</td>
<td>Cross database object found &amp; skipped in exporting</td>
</tr>
<tr>
<td>675</td>
<td>ERR_SQL_EXST_GRAPH_WORKSPACE</td>
<td>Cannot use duplicate graph workspace name</td>
</tr>
<tr>
<td>676</td>
<td>ERR_SQL_DUP_WORKLOAD_MAPPING</td>
<td>Cannot use duplicate combination of workload mapping</td>
</tr>
<tr>
<td>677</td>
<td>ERR_SQL_CHECK_CONSTRAINT_VIOLATION</td>
<td>Check constraint violation</td>
</tr>
<tr>
<td>678</td>
<td>ERR_SQL_PLANSTABILIZER_DEPRECATED</td>
<td>Deprecated plan stabilizer error</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>679</td>
<td>ERR_SQL_PLANSTABILIZER_NO_MANAGER_DEPRECATED</td>
<td>Deprecated plan stabilizer error - manager not found: please check if Plan Stabilizer is enabled</td>
</tr>
<tr>
<td>680</td>
<td>ERR_SQL_STATEMENT_HINT</td>
<td>Statement hint error</td>
</tr>
<tr>
<td>681</td>
<td>ERR_SQL_STATEMENT_HINT_COMMAND</td>
<td>Statement hint error - error while processing statement hint command</td>
</tr>
<tr>
<td>682</td>
<td>ERR_SQL_STATEMENT_HINT_TABLE_EMPTY</td>
<td>Statement hint error - statement hint table is empty</td>
</tr>
<tr>
<td>683</td>
<td>ERR_SQL_STATEMENT_HINT_TABLE_MAP_LOAD_ERROR</td>
<td>Statement hint error - statement hint table is corrupt</td>
</tr>
<tr>
<td>684</td>
<td>ERR_SQL_STATEMENT_HINT_RECORD_ALREADY_EXISTS</td>
<td>Statement hint error - statement hint record already exists</td>
</tr>
<tr>
<td>685</td>
<td>ERR_SQL_STATEMENT_HINT_RECORD_DOES_NOT_EXIST</td>
<td>Statement hint error - statement hint record does not exist</td>
</tr>
<tr>
<td>686</td>
<td>ERR_SQL_START_TASK_ERROR</td>
<td>Start task error</td>
</tr>
<tr>
<td>687</td>
<td>ERR_SQL_EXCEED_LAG_TIME</td>
<td>Exceed lag time of RESULT_LAG</td>
</tr>
<tr>
<td>688</td>
<td>ERR_IO_FAILURE_ON_FILE_WRITE</td>
<td>I/O error occurred on file write</td>
</tr>
<tr>
<td>689</td>
<td>ERR_SQL_DUPLICATE_ROWID_MATCHED</td>
<td>Duplicate rowid matched during merge into</td>
</tr>
<tr>
<td>690</td>
<td>ERR_SQL_PLAN_STABILITY_DEPRECATED</td>
<td>Deprecated plan stability error</td>
</tr>
<tr>
<td>691</td>
<td>ERR_SQL_PLAN_STABILITY_COMMAND_DEPRECATED</td>
<td>Deprecated plan stability error - error while processing command</td>
</tr>
<tr>
<td>692</td>
<td>ERR_SQL_PLAN_STABILITY_TABLE_EMPTY_DEPRECATED</td>
<td>Deprecated plan stability error - stored plan table is empty</td>
</tr>
<tr>
<td>693</td>
<td>ERR_SQL_PLAN_STABILITY_MAP_LOAD_ERROR_DEPRECATED</td>
<td>Deprecated plan stability error - stored plan table is corrupt</td>
</tr>
<tr>
<td>694</td>
<td>ERR_SQL_PLAN_STABILITY_RECORD_ALREADY_EXISTS_DEPRECATED</td>
<td>Deprecated plan stability error - stored plan record already exists</td>
</tr>
<tr>
<td>695</td>
<td>ERR_SQL_PLAN_STABILITY_RECORD_DOES_NOT_EXIST_DEPRECATED</td>
<td>Deprecated plan stability error - stored plan record does not exist</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>696</td>
<td>ERR_SQL_PLAN_STABILITY_CANNOT_CONVERT_ABSTRACT_PLAN_DEPRECATED</td>
<td>Deprecated plan stability error - cannot convert to abstract plan</td>
</tr>
<tr>
<td>697</td>
<td>ERR_SQL_PREACTIVE_KEY_EXISTS</td>
<td>Preactive key already exists</td>
</tr>
<tr>
<td>698</td>
<td>ERR_SQL_NO_PREACTIVE_KEY</td>
<td>No preactive key exists</td>
</tr>
<tr>
<td>699</td>
<td>ERR_SQL_EXST_DEPENDENCY_RULE</td>
<td>Cannot use duplicate dependency rule name</td>
</tr>
<tr>
<td>700</td>
<td>ERR_SQL_SINGLE_COLUMN_SEARCH_THROW_ERROR</td>
<td>No_stacked_column_search(throw_error) error</td>
</tr>
<tr>
<td>701</td>
<td>ERR_SQL_EXST_USERGROUP</td>
<td>Usergroup name already exists</td>
</tr>
<tr>
<td>702</td>
<td>ERR_SQL_INV_USERGROUP</td>
<td>Invalid usergroup name</td>
</tr>
<tr>
<td>703</td>
<td>ERR_INCORRECT_ROOT_KEYS_BACKUP_PASSWORD</td>
<td>Incorrect root keys backup password</td>
</tr>
<tr>
<td>704</td>
<td>ERR_SQL_USERGROUP_DELETION_FAILED</td>
<td>Usergroup cannot be dropped</td>
</tr>
<tr>
<td>705</td>
<td>ERR_SQL_CONCURRENT_GRANT</td>
<td>Two concurrent statements performed the same grant operation</td>
</tr>
<tr>
<td>706</td>
<td>ERR_SQL_INV_SYMMETRIC_CIPHER</td>
<td>Currently only AES-256-CBC is supported: invalid cipher</td>
</tr>
<tr>
<td>707</td>
<td>ERR_SQL_EXST_COLUMN_KEY</td>
<td>Cannot use duplicate column key name</td>
</tr>
<tr>
<td>708</td>
<td>ERR_SQL_EXST_COLUMN_KEYCOPY</td>
<td>Column keycopy already exists</td>
</tr>
<tr>
<td>709</td>
<td>ERR_SQL_EXST_KEYPAIR</td>
<td>Keypair already exists</td>
</tr>
<tr>
<td>710</td>
<td>ERR_SQL_INVASYMMETRIC_CIPHER</td>
<td>Currently only RSA-OAEP-2048 is supported: invalid cipher</td>
</tr>
<tr>
<td>711</td>
<td>ERR_SQL_EXST_COLUMN_KEY_ID</td>
<td>Cannot use duplicate column key id</td>
</tr>
<tr>
<td>712</td>
<td>ERR_SQL_PLAN_STABILITY_MIGRATION_DEPRECATED</td>
<td>Deprecated plan stabilizer stored plan error - migration error</td>
</tr>
<tr>
<td>713</td>
<td>ERR_SQL_NOT_OWN_KEYPAIR</td>
<td>Keypair not owned by the creator of the column key</td>
</tr>
<tr>
<td>714</td>
<td>ERR_SQL_DROP_COLUMN_KEYCOPY</td>
<td>Cannot drop the last key admin key-copy</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>715</td>
<td>ERR_SQL_EMPTY_WORKLOAD_MAPPING</td>
<td>Cannot use a workload mapping with no properties</td>
</tr>
<tr>
<td>716</td>
<td>ERR_SQL_STALE_STATEMENT</td>
<td>Statement is stale</td>
</tr>
<tr>
<td>717</td>
<td>ERR_SQL_INV_KEY_ID</td>
<td>Invalid key id</td>
</tr>
<tr>
<td>718</td>
<td>ERR_SQL_CANNOT_UPDATE_NOT_EXISTING_ROW</td>
<td>Cannot update non-existent row</td>
</tr>
<tr>
<td>719</td>
<td>ERR_SQL_CURRENCY_UNIT_CONVERSION</td>
<td>Currency/unit conversion error</td>
</tr>
<tr>
<td>720</td>
<td>ERR_SQL_DDL_DISABLED</td>
<td>DDL is currently disabled in this session</td>
</tr>
<tr>
<td>721</td>
<td>ERR_SQL_MISSING_ROWID</td>
<td>Rowid column is not found</td>
</tr>
<tr>
<td>722</td>
<td>ERR_SQL_DROP_LAST_COLUMN_KEYCOPY</td>
<td>Cannot drop the last keycopy</td>
</tr>
<tr>
<td>723</td>
<td>ERR_SQL_NOT_KEY_ADMIN_KEYPAIR</td>
<td>Keypair not owned by key admin</td>
</tr>
<tr>
<td>724</td>
<td>ERR_SQL_INV_LIBRARY</td>
<td>Invalid name of library</td>
</tr>
<tr>
<td>725</td>
<td>ERR_SQL_INV_USABLE_LIBRARY</td>
<td>Invalidated library</td>
</tr>
<tr>
<td>726</td>
<td>ERR_SQL_INVALID_BASE64_STRING</td>
<td>Invalid Base64 string</td>
</tr>
<tr>
<td>727</td>
<td>ERR_SQL_INV_CACHE</td>
<td>Invalid result cache name</td>
</tr>
<tr>
<td>728</td>
<td>ERR_SQL_EXST_CACHE</td>
<td>Cannot use duplicate result cache name</td>
</tr>
<tr>
<td>729</td>
<td>ERR_SQL_UNKNOWN_SESSION</td>
<td>Unknown session id</td>
</tr>
<tr>
<td>730</td>
<td>ERR_SQL_RECOMPILE_WITHOUT_FALLBACK</td>
<td>Query recompilation is required for re-optimized plan generation</td>
</tr>
<tr>
<td>731</td>
<td>ERR_SQL_RECOMPILE_WITH_FALLBACK</td>
<td>Query recompilation is required for legacy plan generation</td>
</tr>
<tr>
<td>732</td>
<td>ERR_SQL_FAIL_AUTO_MATCH_CACHE</td>
<td>Cannot find matching view</td>
</tr>
<tr>
<td>733</td>
<td>ERR_SQL_DDL_ONLY_OBJECT</td>
<td>No DML on DDL-only object</td>
</tr>
<tr>
<td>734</td>
<td>ERR_SQL_EMPTY_COLUMN_KEY</td>
<td>Cannot use empty column encryption key</td>
</tr>
<tr>
<td>735</td>
<td>ERR_SQL_RECOMPILE_WITH_DISK</td>
<td>Query recompilation is required for disk plan generation</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>736</td>
<td>ERR_SQL_RE-MOVE_SQL_PLAN_CACHE_SELECTIVE</td>
<td>Invalid WHERE condition.</td>
</tr>
<tr>
<td>737</td>
<td>ERR_SQL FUZZY_SEARCH_INDEX</td>
<td>Cannot create fuzzy search index</td>
</tr>
<tr>
<td>738</td>
<td>ERR_SQL_INV_RESULT_CACHE_TYPE</td>
<td>Invalid result cache type</td>
</tr>
<tr>
<td>1024</td>
<td>ERR SES</td>
<td>Session error</td>
</tr>
<tr>
<td>1025</td>
<td>ERR COM</td>
<td>Communication error</td>
</tr>
<tr>
<td>1026</td>
<td>ERR COM LISTEN</td>
<td>Cannot bind a communication port</td>
</tr>
<tr>
<td>1027</td>
<td>ERR COM INIT</td>
<td>Communication initialization error</td>
</tr>
<tr>
<td>1028</td>
<td>ERR COM_IOCTL</td>
<td>I/O control error</td>
</tr>
<tr>
<td>1029</td>
<td>ERR COM CONNECT</td>
<td>Connection failure</td>
</tr>
<tr>
<td>1030</td>
<td>ERR COM SEND</td>
<td>Send error</td>
</tr>
<tr>
<td>1031</td>
<td>ERR COM RECEIVE</td>
<td>Receive error</td>
</tr>
<tr>
<td>1032</td>
<td>ERR SES THREAD CREATE</td>
<td>Cannot create a thread</td>
</tr>
<tr>
<td>1033</td>
<td>ERR SES INV PROTOCOL</td>
<td>Error while parsing protocol</td>
</tr>
<tr>
<td>1034</td>
<td>ERR SES EXCEED MAX SESSION</td>
<td>Exceed maximum number of sessions</td>
</tr>
<tr>
<td>1035</td>
<td>ERR SES INV VERSION</td>
<td>Not supported version</td>
</tr>
<tr>
<td>1036</td>
<td>ERR SES INV SESSION</td>
<td>Invalid session id</td>
</tr>
<tr>
<td>1037</td>
<td>ERR COM UNKNOWN HOST</td>
<td>Unknown hostname</td>
</tr>
<tr>
<td>1038</td>
<td>ERR SES SERVER BUSY</td>
<td>Rejected as server is temporarily overloaded</td>
</tr>
<tr>
<td>1088</td>
<td>ERR DATA STAT</td>
<td>Data statistics error</td>
</tr>
<tr>
<td>1089</td>
<td>ERR DATA STAT NOT FOUND</td>
<td>No matching data statistics objects found</td>
</tr>
<tr>
<td>1090</td>
<td>ERR DATA STAT REMOTE_QUERY_ERR</td>
<td>Invalid result from query to remote source</td>
</tr>
<tr>
<td>1091</td>
<td>ERR DATA STAT TABLE NOT FOUND</td>
<td>Specified table not found or not supported</td>
</tr>
<tr>
<td>1092</td>
<td>ERR DATA STAT BUILD_ERROR</td>
<td>Error building data statistics object</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>1093</td>
<td>ERR_DATA_STAT_EXISTS</td>
<td>Data statistics object already exists</td>
</tr>
<tr>
<td>1094</td>
<td>ERR_DATA_STAT_INVALID_SETTING</td>
<td>Invalid combination of settings specified</td>
</tr>
<tr>
<td>1120</td>
<td>ERR_USERGROUP_GENERAL</td>
<td>Usergroup error</td>
</tr>
<tr>
<td>1121</td>
<td>ERR_USERGROUP_USER_NOT_MEMBER_OF_ANY</td>
<td>User is not member of any usergroup</td>
</tr>
<tr>
<td>1122</td>
<td>ERR_USERGROUP_EQUAL_CURRENT_AND_NEW_USERGROUP</td>
<td>Current and new usergroup are the same</td>
</tr>
<tr>
<td>1123</td>
<td>ERR_USERGROUP_UNKNOWN_PARAMETER_NAME</td>
<td>Unknown parameter for usergroup</td>
</tr>
<tr>
<td>1124</td>
<td>ERR_USERGROUP_UNKNOWN_PARAMETER_SET_NAME</td>
<td>Unknown parameter set for usergroup</td>
</tr>
<tr>
<td>1125</td>
<td>ERR_USERGROUP_DUPLICATE_PARAMETER_NAME</td>
<td>Same parametername specified more than once</td>
</tr>
<tr>
<td>1126</td>
<td>ERR_USERGROUP_INVALID_PARAMETER_VALUE</td>
<td>Invalid value for usergroup parameter</td>
</tr>
<tr>
<td>1127</td>
<td>ERR_USERGROUP_DISABLE_CONNECT_BY_GROUP_MEMBER_NOT_ALLOWED</td>
<td>User is member of this usergroup</td>
</tr>
<tr>
<td>1152</td>
<td>ERR_SQL_PLAN_STABILITY</td>
<td>Plan stability error</td>
</tr>
<tr>
<td>1153</td>
<td>ERR_SQL_PLAN_STABILITY_COMMAND</td>
<td>Plan stability error - cannot execute command</td>
</tr>
<tr>
<td>1154</td>
<td>ERR_SQL_PLAN_STABILITY_TABLE_EMPTY</td>
<td>Plan stability error - abstract sql plan table is empty</td>
</tr>
<tr>
<td>1155</td>
<td>ERR_SQL_PLAN_STABILITY_MAP_LOAD_ERROR</td>
<td>Plan stability error - cannot load abstract sql plan</td>
</tr>
<tr>
<td>1156</td>
<td>ERR_SQL_PLAN_STABILITY_RECORD_ALREADY_EXISTS</td>
<td>Plan stability error - abstract sql plan record already exists</td>
</tr>
<tr>
<td>1157</td>
<td>ERR_SQL_PLAN_STABILITY_RECORD_DOES_NOT_EXIST</td>
<td>Plan stability error - abstract sql plan record does not exist</td>
</tr>
<tr>
<td>1158</td>
<td>ERR_SQL_PLAN_STABILITY_CANNOT_CAPTURE</td>
<td>Cannot capture abstract sql plan</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>1159</td>
<td>ERR_SQL_PLAN_STABILITY_MIGRATION</td>
<td>Plan stability error - migration</td>
</tr>
<tr>
<td>1160</td>
<td>ERR_SQL_PLAN_STABILITY_CANNOT_APPLY</td>
<td>Cannot apply abstract sql plan</td>
</tr>
<tr>
<td>1161</td>
<td>ERR_SQL_PLAN_STABILITY_JSON_ERROR</td>
<td>Cannot deserialize abstract sql plan</td>
</tr>
<tr>
<td>1162</td>
<td>ERR_SQL_PLAN_STABILITY_UPDATE_LOCATION</td>
<td>Plan stability error - update location first</td>
</tr>
<tr>
<td>1163</td>
<td>ERR_SQL_PLAN_STABILITY_RELATED_OBJECT_CHECK_FAILED</td>
<td>Plan stability error - related object mismatch</td>
</tr>
<tr>
<td>1280</td>
<td>ERR_SQLSCRIPT_2</td>
<td>Sqlscript error</td>
</tr>
<tr>
<td>1281</td>
<td>ERR_SQLSCRIPT_WRONG_PARAMS</td>
<td>Wrong number or types of parameters in call</td>
</tr>
<tr>
<td>1282</td>
<td>ERR_SQLSCRIPT_OUT_PARAM_VAR</td>
<td>Output parameter not a variable</td>
</tr>
<tr>
<td>1283</td>
<td>ERR_SQLSCRIPT_OUT_PARAM_DEFAULT</td>
<td>OUT and IN OUT parameters may not have default expresions</td>
</tr>
<tr>
<td>1284</td>
<td>ERR_SQLSCRIPT_DUP_PARAMETERS</td>
<td>Duplicate parameters are not permitted</td>
</tr>
<tr>
<td>1285</td>
<td>ERR_SQLSCRIPT_DUP_DECL</td>
<td>At most one declaration is permitted in the declaration section</td>
</tr>
<tr>
<td>1286</td>
<td>ERR_SQLSCRIPT_CURSOR_SELECT_STMT</td>
<td>Cursor must be declared by SELECT statement</td>
</tr>
<tr>
<td>1287</td>
<td>ERR_SQLSCRIPT_ID_NOT_DECLARED</td>
<td>Identifier must be declared</td>
</tr>
<tr>
<td>1288</td>
<td>ERR_SQLSCRIPT_NOT_ASSIGN_TARGET</td>
<td>Expression cannot be used as an assignment target</td>
</tr>
<tr>
<td>1289</td>
<td>ERR_SQLSCRIPT_NOT INTO_TARGET</td>
<td>Not allowed expression for INTO-target</td>
</tr>
<tr>
<td>1290</td>
<td>ERR_SQLSCRIPT_LHS_CANNOT_ASSIGNED</td>
<td>Expression is inappropriate as the left hand side of an assignment statement</td>
</tr>
<tr>
<td>1291</td>
<td>ERR_SQLSCRIPT_EXPR_WRONG_TYPE</td>
<td>Expression is of wrong type</td>
</tr>
<tr>
<td>1292</td>
<td>ERR_SQLSCRIPT_ILLEGAL_EXIT_STMT</td>
<td>Illegal loop control statement</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>1293</td>
<td>ERR_SQLSCRIPT_ID_EXCEPTION_TYPE</td>
<td>Not a condition variable</td>
</tr>
<tr>
<td>1294</td>
<td>ERR_SQLSCRIPT_INTO_CLAUSE</td>
<td>An INTO clause is expected in SELECT statement</td>
</tr>
<tr>
<td>1295</td>
<td>ERR_SQLSCRIPT_NOT_ALLOWED_STMT</td>
<td>Not permitted statement</td>
</tr>
<tr>
<td>1296</td>
<td>ERR_SQLSCRIPT_NOT_CURSOR</td>
<td>Identifier is not a cursor</td>
</tr>
<tr>
<td>1297</td>
<td>ERR_SQLSCRIPT_NUM_FETCH_VALUES</td>
<td>Wrong number of values in the INTO list of a FETCH statement</td>
</tr>
<tr>
<td>1298</td>
<td>ERR_SQLSCRIPT_UNHANDLED_EXCEPTION</td>
<td>Unhandled user-defined exception</td>
</tr>
<tr>
<td>1299</td>
<td>ERR_SQLSCRIPT_NO_DATA_FOUND</td>
<td>No data found</td>
</tr>
<tr>
<td>1300</td>
<td>ERR_SQLSCRIPT_FETCH_MANY_ROWS</td>
<td>Fetch returns more than requested number of rows</td>
</tr>
<tr>
<td>1301</td>
<td>ERR_SQLSCRIPT_VALUE_ERROR</td>
<td>Numeric or value error</td>
</tr>
<tr>
<td>1302</td>
<td>ERR_SQLSCRIPT_OUT_PARAM_IN_FUNCTION</td>
<td>Parallelizable function cannot have OUT or IN OUT parameter</td>
</tr>
<tr>
<td>1303</td>
<td>ERR_SQLSCRIPT_USER_DEFINED_EXCEPTION</td>
<td>User-defined exception</td>
</tr>
<tr>
<td>1304</td>
<td>ERR_SQLSCRIPT_CURSOR_ALREADY_OPEN</td>
<td>Cursor is already opened</td>
</tr>
<tr>
<td>1305</td>
<td>ERR_SQLSCRIPT_INVALID_RETURN_TYPE</td>
<td>Return type is invalid</td>
</tr>
<tr>
<td>1306</td>
<td>ERR_SQLSCRIPT_RETURN_TYPE_MISMATCH</td>
<td>Return type mismatch</td>
</tr>
<tr>
<td>1307</td>
<td>ERR_SQLSCRIPT_UNSUPPORTED_DATATYPE</td>
<td>Unsupported datatype is used</td>
</tr>
<tr>
<td>1308</td>
<td>ERR_SQLSCRIPT_INVALID_SINGLE_ASSIGNMENT</td>
<td>Illegal single assignment</td>
</tr>
<tr>
<td>1309</td>
<td>ERR_SQLSCRIPT_INVALID_USE_OF_TABLE_VARIABLE</td>
<td>Invalid use of table variable</td>
</tr>
<tr>
<td>1310</td>
<td>ERR_SQLSCRIPT_NOT_ALLOWED_SCALAR_TYPE</td>
<td>Scalar type is not allowed</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>1311</td>
<td>ERR_SQLSCRIPT_NO_OUT_PARAM</td>
<td>Out parameter is not specified</td>
</tr>
<tr>
<td>1312</td>
<td>ERR_SQLSCRIPT_AT_MOST_ONE_OUT_PARAM</td>
<td>At most one output parameter is allowed</td>
</tr>
<tr>
<td>1313</td>
<td>ERR_SQLSCRIPT_OUT_PARAM_TABLE</td>
<td>Output parameter should be a table or a table variable</td>
</tr>
<tr>
<td>1314</td>
<td>ERR_SQLSCRIPT_INVALID_VARIABLE_NAME</td>
<td>Invalid or reserved variable name</td>
</tr>
<tr>
<td>1315</td>
<td>ERR_SQLSCRIPT_RETURN_RESULT_SET_WITH_RESULTVIEW</td>
<td>Return result set from SELECT statement exist when result view is defined</td>
</tr>
<tr>
<td>1316</td>
<td>ERR_SQLSCRIPT_NOT_ASSIGNED_OUT_TABVAR</td>
<td>Some out table variable is not assigned</td>
</tr>
<tr>
<td>1317</td>
<td>ERR_SQLSCRIPT_FUNCTION_NAME_MAX_LEN</td>
<td>Function name exceeds maximum length limitation</td>
</tr>
<tr>
<td>1318</td>
<td>ERR_SQLSCRIPT_BUILTIN_NOT_DEFINED</td>
<td>Built-in function not defined</td>
</tr>
<tr>
<td>1319</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_TABLE_NAME</td>
<td>Parameter must be a table name</td>
</tr>
<tr>
<td>1320</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_ATTRIBUTE_WITH_SCHEMA</td>
<td>Parameter must be an attribute name without a table name upfront</td>
</tr>
<tr>
<td>1321</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_ATTRIBUTE_WITH_ALIAS</td>
<td>Parameter must be an attribute name without an alias</td>
</tr>
<tr>
<td>1322</td>
<td>ERR_SQLSCRIPT_CALC_ATTR_NOT_ALLOWED</td>
<td>CE_CALC not allowed</td>
</tr>
<tr>
<td>1323</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_COL_OR_AGGR_VECTOR</td>
<td>Parameter must be a vector of columns or aggregations</td>
</tr>
<tr>
<td>1324</td>
<td>ERR_SQLSCRIPT_BUILTIN_MISSING_JOIN_ATTR_IN_PROJECTION</td>
<td>Join attribute must be available in projection list</td>
</tr>
<tr>
<td>1325</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_SQLIDENT_VECTOR</td>
<td>Parameter must be a vector of sql identifiers</td>
</tr>
<tr>
<td>1326</td>
<td>ERR_SQLSCRIPT_DUPLICATE_ATTRIBUTE_NAME</td>
<td>Duplicate attribute name</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1327</td>
<td>ERR_SQLSCRIPT_PARAM_UNSUPPORTED_TYPE</td>
<td>Parameter has a non supported type</td>
</tr>
<tr>
<td>1328</td>
<td>ERR_SQLSCRIPT_BUILTIN_MISSING_ATTRIBUTE_IN_PROJECTION</td>
<td>Attribute not found in column table</td>
</tr>
<tr>
<td>1329</td>
<td>ERR_SQLSCRIPT_BUILTIN_DUPLICATE_COLUMN_NAME</td>
<td>Duplicate column name</td>
</tr>
<tr>
<td>1330</td>
<td>ERR_SQLSCRIPT_BUILTIN_CALCATTR_EXPRESSION_SYNTAX</td>
<td>Syntax Error for calculated Attribute</td>
</tr>
<tr>
<td>1331</td>
<td>ERR_SQLSCRIPT_BUILTIN_FILTER_EXPRESSION_SYNTAX</td>
<td>Syntax Error in filter expression</td>
</tr>
<tr>
<td>1332</td>
<td>ERR_SQLSCRIPT_BUILTIN_FIRST_PARAM_NOT_COLUMN_TABLE</td>
<td>Parameter must be a valid column table or projection view on column tables</td>
</tr>
<tr>
<td>1333</td>
<td>ERR_SQLSCRIPT_BUILTIN_JOIN_ATTR_NOT_FOUND_IN_VAR</td>
<td>Join attributes not found in variable</td>
</tr>
<tr>
<td>1334</td>
<td>ERR_SQLSCRIPT_BUILTIN_IN_PARAM_NOTSAME_TABLE_TYPE</td>
<td>Input parameters do not have the same table type</td>
</tr>
<tr>
<td>1335</td>
<td>ERR_SQLSCRIPT_RUNTIME_CYCLIC_DEPENDENCY</td>
<td>Cyclic dependency found in a runtime procedure</td>
</tr>
<tr>
<td>1336</td>
<td>ERR_SQLSCRIPT_RUNTIME_UNEXPECTED_EXCEPTION</td>
<td>Unexpected internal exception caught in a runtime procedure</td>
</tr>
<tr>
<td>1337</td>
<td>ERR_SQLSCRIPT_VAR_DEPENDS_ON_UNDEFINED_VARIABLE</td>
<td>Variable depends on an unassigned variable</td>
</tr>
<tr>
<td>1338</td>
<td>ERR_SQLSCRIPT_CE_CONVERSION_CUSTOM_TAB_MISSING</td>
<td>Cannot find table used in CE_CONVERSION</td>
</tr>
<tr>
<td>1339</td>
<td>ERR_SQLSCRIPT_TOO_MANY_PARAMS</td>
<td>Too many parameters</td>
</tr>
<tr>
<td>1340</td>
<td>ERR_SQLSCRIPT_NESTED_CALL_TOO_DEEP</td>
<td>The depth of the nested call is too deep</td>
</tr>
<tr>
<td>1341</td>
<td>ERR_SQLSCRIPT_VERSION_VALIDATION_FAILED</td>
<td>Procedure version validation failed</td>
</tr>
<tr>
<td>1342</td>
<td>ERR_SQLSCRIPT_CE_CALC_ATTRIBUTE_AND_ALIAS_ARE_SAME</td>
<td>Attribute has the same name as the alias</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1343</td>
<td>ERR_SQLSCRIPT_RETRY_EXCEPTION</td>
<td>Retry Exception is occurred in a runtime procedure</td>
</tr>
<tr>
<td>1344</td>
<td>ERR_SQLSCRIPT_NOT_ALLOWED_DYNAMIC_SQL</td>
<td>Dynamic SQL or DDL is not allowed</td>
</tr>
<tr>
<td>1345</td>
<td>ERR_SQLSCRIPT_NOT_ALLOWED_CONCURRENT_WRITES</td>
<td>Not allowed concurrent write</td>
</tr>
<tr>
<td>1346</td>
<td>ERR_SQLSCRIPT_NOT_ALLOWED_CONCURRENT_READ_AND_WRITE</td>
<td>Not allowed concurrent read and write</td>
</tr>
<tr>
<td>1347</td>
<td>WRN_SQLSCRIPT_NOT_RECOMMENDED_FEATURE</td>
<td>Not recommended feature</td>
</tr>
<tr>
<td>1348</td>
<td>ERR_SQLSCRIPT_LLANG_GET_LIBRARY_IMPORT_LIST_FAILED</td>
<td>Failed to retrieve the list of imported libraries from LLANG procedure</td>
</tr>
<tr>
<td>1349</td>
<td>ERR_SQLSCRIPT_INITIAL_ASSIGNMENT_REQUIRED_FOR_CONSTANT_TABLE</td>
<td>Assigning initial value is required for declaring constant table variable</td>
</tr>
<tr>
<td>1350</td>
<td>ERR_SQLSCRIPT_NOT_ALLOWED_NON_DETERMINISTIC_FEATURE</td>
<td>Non-deterministic feature is not allowed</td>
</tr>
<tr>
<td>1351</td>
<td>ERR_SQLSCRIPT_INVALID_PARSE_TREE</td>
<td>Invalid parse tree</td>
</tr>
<tr>
<td>1352</td>
<td>ERR_SQLSCRIPT_ENCRYPTION_NOT_ALLOWED</td>
<td>Not allowed for encrypted procedure or function</td>
</tr>
<tr>
<td>1353</td>
<td>ERR_SQLSCRIPT_NOT_NULL_COLUMN_IGNORED</td>
<td>NOT NULL constraints in explicit table types are ignored</td>
</tr>
<tr>
<td>1354</td>
<td>ERR_SQLSCRIPT_CURSOR_NOT_OPENED</td>
<td>Cursor to be fetched has not been opened yet</td>
</tr>
<tr>
<td>1355</td>
<td>ERR_SQLSCRIPT_INVALID_EXTERNAL_LANG</td>
<td>Invalid external language</td>
</tr>
<tr>
<td>1356</td>
<td>ERR_SQLSCRIPT_COMPOSITE</td>
<td>Composite error in SQLScript processing</td>
</tr>
<tr>
<td>1357</td>
<td>ERR_SQLSCRIPT_CE_TYPE_MISMATCH</td>
<td>Datatype mismatch in CE function</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>1358</td>
<td>ERR_SQLSCRIPT_LIB_WITHOUT_USING</td>
<td>Require USING statement before use</td>
</tr>
<tr>
<td>1359</td>
<td>ERR_SQLSCRIPT_TOO_MANY_MEMBERS</td>
<td>Surpassed expected library member count</td>
</tr>
<tr>
<td>1360</td>
<td>ERR_SQLSCRIPT_INVALID_INPUT</td>
<td>Invalid input</td>
</tr>
<tr>
<td>1361</td>
<td>ERR_SQLSCRIPT_INVALID_PRAGMA</td>
<td>Invalid pragma</td>
</tr>
<tr>
<td>1362</td>
<td>ERR_SQLSCRIPT_CALL_STACK_TOO_DEEP</td>
<td>The depth of the sqlscript callstack is too deep</td>
</tr>
<tr>
<td>1363</td>
<td>ERR_SQLSCRIPT_USER_CONDITION</td>
<td>User-defined condition</td>
</tr>
<tr>
<td>1364</td>
<td>ERR_SQLSCRIPT_ASYNC_CALL_INIT_FAILED</td>
<td>Asynchronous procedure call initialization failed</td>
</tr>
<tr>
<td>1365</td>
<td>ERR_SQLSCRIPT_ASYNC_CALL_CANCELLATION_FAILED</td>
<td>Asynchronous procedure call cancellation failed</td>
</tr>
<tr>
<td>1536</td>
<td>ERR_STATISTICS_MONITOR_INTERNAL_ERROR</td>
<td>System monitor internal error</td>
</tr>
<tr>
<td>1776</td>
<td>ERR_CONFIG_UNSUPPORTED</td>
<td>Configuration error:</td>
</tr>
<tr>
<td>1777</td>
<td>WRN_CONFIG_UNSUPPORTED</td>
<td>Configuration warning:</td>
</tr>
<tr>
<td>1792</td>
<td>ERR_SHM</td>
<td>Shared memory error</td>
</tr>
<tr>
<td>1793</td>
<td>ERR_SHM_CREATE_INVALID</td>
<td>Invalid key or invalid size</td>
</tr>
<tr>
<td>1794</td>
<td>ERR_SHM_CREATE_ALREADY_EXIST</td>
<td>The segment already exists</td>
</tr>
<tr>
<td>1795</td>
<td>ERR_SHM_CREATE_EXCEED_LIMIT</td>
<td>Exceed the system-wide limit on shared memory</td>
</tr>
<tr>
<td>1796</td>
<td>ERR_SHM_CREATE_NOT_EXIST</td>
<td>No segment exists for the given key</td>
</tr>
<tr>
<td>1797</td>
<td>ERR_SHM_CREATE_NO_ACCESS</td>
<td>The user does not have permission to access the shared memory segment</td>
</tr>
<tr>
<td>1798</td>
<td>ERR_SHM_CREATE_NO_MORE_MEMORY</td>
<td>No memory could be allocated for segment overhead</td>
</tr>
<tr>
<td>1799</td>
<td>ERR_SHM_DROP_INVALID</td>
<td>Invalid shmid</td>
</tr>
<tr>
<td>1800</td>
<td>ERR_SHM_DROP_NO_ACCESS</td>
<td>Allow read access for shmid</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1801</td>
<td>ERR_SHM_DROP_REMOVED_IDENTIFIER</td>
<td>Shmid points to a removed identifier</td>
</tr>
<tr>
<td>1802</td>
<td>ERR_SHM_DROP_NO_PERMISSION</td>
<td>The effective user ID of the calling process is not the creator</td>
</tr>
<tr>
<td>1803</td>
<td>ERR_SHM_DROP_OVERFLOW</td>
<td>The gid or uid value is too large to be stored in the structure</td>
</tr>
<tr>
<td>1804</td>
<td>ERR_SHM_ATTACH_NO_ACCESS</td>
<td>The user does not have permission to access the shared memory segment</td>
</tr>
<tr>
<td>1805</td>
<td>ERR_SHM_ATTACH_INVALID</td>
<td>Invalid shmid</td>
</tr>
<tr>
<td>1806</td>
<td>ERR_SHM_ATTACH_NO_MORE_MEMORY</td>
<td>No memory could be allocated for the descriptor or for the page tables</td>
</tr>
<tr>
<td>1807</td>
<td>ERR_SHM_UNKNOWN</td>
<td>Unknown shared memory error</td>
</tr>
<tr>
<td>1856</td>
<td>ERR_SQL_PLAN_EXECUTION_MONITOR_GENERAL</td>
<td>[SQL Plan Execution Monitor] general error</td>
</tr>
<tr>
<td>2048</td>
<td>ERR_CS</td>
<td>Column store error</td>
</tr>
<tr>
<td>2049</td>
<td>ERR_CS_NO_PRIMARY_KEY</td>
<td>Primary key is not specified for column table</td>
</tr>
<tr>
<td>2050</td>
<td>ERR_CS_NOT_SUPPORTED_DDL</td>
<td>Not supported DDL type for column table</td>
</tr>
<tr>
<td>2051</td>
<td>ERR_CS_NOT_SUPPORTED_DATA_TYPE</td>
<td>Not supported data type for column table</td>
</tr>
<tr>
<td>2052</td>
<td>ERR_CS_NOT_SUPPORTED_DML</td>
<td>Not supported DML type for column table</td>
</tr>
<tr>
<td>2053</td>
<td>ERR_CS_INVALID_RETURNED_VALUE</td>
<td>Invalid returned value from attribute engine</td>
</tr>
<tr>
<td>2054</td>
<td>ERR_CS_DELTA_LOG_REPLAY_FAILED</td>
<td>Delta log replay failed</td>
</tr>
<tr>
<td>2055</td>
<td>ERR_CS_MAXIMUM_ROW</td>
<td>Maximum number of rows per table or partition reached</td>
</tr>
<tr>
<td>2056</td>
<td>ERR_CS_CONSISTENCY_CHECK_FAILED</td>
<td>Column store table consistency check failed</td>
</tr>
<tr>
<td>2304</td>
<td>ERR_PYDBAPI</td>
<td>Python dbapi error</td>
</tr>
<tr>
<td>2305</td>
<td>ERR_PYDBAPI_INTERFACE_FAILURE</td>
<td>Interface failure</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>2306</td>
<td>ERR_PYDBAPI_PROGRAMMING_MISTAKE</td>
<td>Programming mistake</td>
</tr>
<tr>
<td>2307</td>
<td>ERR_PYDBAPI_INVALID_QUERY_PARAMETER</td>
<td>Invalid query parameter</td>
</tr>
<tr>
<td>2308</td>
<td>ERR_PYDBAPI_NOT_SUPPORTED_STR_ENCODING</td>
<td>Not supported encoding for string</td>
</tr>
<tr>
<td>2560</td>
<td>ERR_METADATA</td>
<td>Metadata error</td>
</tr>
<tr>
<td>2561</td>
<td>ERR_DIST_METADATA</td>
<td>Distributed metadata error</td>
</tr>
<tr>
<td>2562</td>
<td>ERR_DIST_METADATA_REDIRECT_FAILURE</td>
<td>DDL redirect error</td>
</tr>
<tr>
<td>2563</td>
<td>ERR_DIST_METADATA_DDL_NOTIFICATION_FAILURE</td>
<td>DDL notification error</td>
</tr>
<tr>
<td>2564</td>
<td>ERR_DIST_METADATA_INVALID_CONTAINER_ID</td>
<td>DDL invalid container id</td>
</tr>
<tr>
<td>2565</td>
<td>ERR_DIST_METADATA_INVALID_INDEX_ID</td>
<td>DDL invalid index id</td>
</tr>
<tr>
<td>2566</td>
<td>ERR_DIST_METADATA_TLSCLIENT_FAILURE</td>
<td>Distributed environment error</td>
</tr>
<tr>
<td>2567</td>
<td>ERR_DIST_METADATA_NETWORK_FAILURE</td>
<td>Network error</td>
</tr>
<tr>
<td>2568</td>
<td>ERR_DIST_METADATA_NOT_SUPPORTED</td>
<td>Metadata update not supported in slave</td>
</tr>
<tr>
<td>2569</td>
<td>ERR_DIST_METADATA_MASTER_UPDATE_FAILURE</td>
<td>Metadata update in master indexserver is failed</td>
</tr>
<tr>
<td>2570</td>
<td>ERR_METADATA_ROW_PAGE_INTEGRITY_FAILURE</td>
<td>Row page consistency check is failed</td>
</tr>
<tr>
<td>2571</td>
<td>ERR_INTEGRITY_BROKEN_METADATA</td>
<td>Integrity check detects broken metadata</td>
</tr>
<tr>
<td>2572</td>
<td>ERR_INTEGRITY_INCONSISTENCY</td>
<td>Integrity check detects inconsistency</td>
</tr>
<tr>
<td>2573</td>
<td>ERR_INTEGRITY_ORPHANED</td>
<td>Integrity check detects orphaned object</td>
</tr>
<tr>
<td>2574</td>
<td>ERR_METADATA_SEGMENT_MIGRATION_GENERAL</td>
<td>Metadata segment migration pre-check error</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2575</td>
<td>ERR_METADATA_SEGMENT_MIGRATION_FATAL</td>
<td>Metadata segment migration internal error</td>
</tr>
<tr>
<td>2576</td>
<td>ERR_CATALOG_INVALID_ADDRESS</td>
<td>Attempt to access metadata of an invalid address (already deleted or corrupted)</td>
</tr>
<tr>
<td>2577</td>
<td>ERR_CATALOG_NULL_TERMINATION</td>
<td>Catalog object has wrong null termination</td>
</tr>
<tr>
<td>2578</td>
<td>ERR_CATALOG_LOCATION_ERROR</td>
<td>Catalog object has wrong location</td>
</tr>
<tr>
<td>2579</td>
<td>ERR_CATALOG_INVALID_REFERENCE</td>
<td>Catalog object has invalid reference (corrupted address)</td>
</tr>
<tr>
<td>2580</td>
<td>ERR_CATALOG_UNEXPECTED_REFERENCE</td>
<td>Catalog object has wrong reference</td>
</tr>
<tr>
<td>2581</td>
<td>ERR_CATALOG_VALUE_DOMAIN_GENERAL</td>
<td>The value of catalog object is broken</td>
</tr>
<tr>
<td>2582</td>
<td>ERR_CATALOG_FOREIGN_KEYCONSTRAINT</td>
<td>Can not find catalog object</td>
</tr>
<tr>
<td>2583</td>
<td>ERR_CATALOG_WRONG_VAR_SLOT_SIZE</td>
<td>Catalog object has wrong slot size</td>
</tr>
<tr>
<td>2584</td>
<td>ERR_CATALOG_CYCLIC_DEPENDENCY</td>
<td>Can not make cyclic dependency</td>
</tr>
<tr>
<td>2585</td>
<td>ERR_ES_BROKEN_METADATA</td>
<td>Inconsistency between HANA and DT catalog</td>
</tr>
<tr>
<td>2586</td>
<td>ERR_ES_METADATA_FIX_FAIL</td>
<td>Repair operation to fix inconsistency failed</td>
</tr>
<tr>
<td>2816</td>
<td>ERR_SQLSCRIPT</td>
<td>SqlScript Error</td>
</tr>
<tr>
<td>2817</td>
<td>ERR_SQLSCRIPT_BUILTIN_TOO_MANY_RETURN_PARAM</td>
<td>SqlScript Builtin Function</td>
</tr>
<tr>
<td>2818</td>
<td>ERR_SQLSCRIPTFUNCTION_NOT_FOUND</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2819</td>
<td>ERR_SQLSCRIPT_TEMPLATE_PARAMETER_NUMBER_WRONG</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2820</td>
<td>ERR_SQLSCRIPT_VARIABLE_NOT_DECLARED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>2821</td>
<td>ERR_SQLSCRIPT_DUPLICATE_VARIABLE_NAME</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2822</td>
<td>ERR_SQLSCRIPT_SQL_EXECUTION_FAILED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2823</td>
<td>ERR_SQLSCRIPT_DROP_FUNCTION_FAILED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2824</td>
<td>ERR_SQLSCRIPT_LOAD_FUNCTION_FAILED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2825</td>
<td>ERR_SQLSCRIPT_SIGNATURE_MISMATCH_WITH_CATALOG</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2826</td>
<td>ERR_SQLSCRIPT_REGISTER_FUNCTION_IN_CATALOG_FAILED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2827</td>
<td>ERR_SQLSCRIPT_SCALAR_INPUT_PARAMS_NOT_SUPPORTED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2828</td>
<td>ERR_SQLSCRIPT_LANGUAGE_NOT_SUPPORTED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2829</td>
<td>ERR_SQLSCRIPT_DROP_FUNCTION_FAILED_EXISTING_CALLER</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2830</td>
<td>ERR_SQLSCRIPT_LLANG_EXACTLY_ONE_OUTPUT_PARAM</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2831</td>
<td>ERR_SQLSCRIPT_BUILTIN_FIRST_PARAM_NOT_COLUMN_TABLE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2832</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_COUNT_NOT_IN_RANGE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2833</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_COUNT_MISMATCH</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2834</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_INPUT</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2835</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_TABLE_NAME</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2836</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_VARIABLE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>2837</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_VARIABLE_VECTOR</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2838</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_SCALAR_VALUE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2839</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_SQLIDENT_VECTOR</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2840</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_ATTRIBUTE_WITH_SCHEMA</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2841</td>
<td>ERR_SQLSCRIPT_BUILTIN_MISSING_ATTRIBUTE_IN_PROJECTION</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2842</td>
<td>ERR_SQLSCRIPT_BUILTIN_MISSING_JOIN_ATTR_IN_PROJECTION</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2843</td>
<td>ERR_SQLSCRIPT_TEMPL_FUNCTION_CANNOT_BE_CALLED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2844</td>
<td>ERR_SQLSCRIPT_PARAM_COUNT_MISMATCH</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2845</td>
<td>ERR_SQLSCRIPT_PARAM_WRONG_TYPE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2846</td>
<td>ERR_SQLSCRIPT_PARAM_WRONG_TYPE_COMPARED_TO_SIGNATURE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2847</td>
<td>ERR_SQLSCRIPT_PARAM_WRONG_TABLE_TYPE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2848</td>
<td>ERR_SQLSCRIPT_PARAM_MODE_MISMATCH</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2849</td>
<td>ERR_SQLSCRIPT_PARAM_UNSUPPORTED_TYPE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2850</td>
<td>ERR_SQLSCRIPT_NO_OUTPUT_PARAM</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2851</td>
<td>ERR_SQLSCRIPT_OUTPUT_PARAM_NOT_TABLE_TYPE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2852</td>
<td>ERR_SQLSCRIPT_BUILTIN_NOT_DEFINED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>Code</td>
<td>Code Description</td>
<td>Type</td>
</tr>
<tr>
<td>------</td>
<td>-----------------</td>
<td>------</td>
</tr>
<tr>
<td>2853</td>
<td>ERR_SQLSCRIPT_VAR_DEPENDS_ON_UNASSIGNED_VAR</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2854</td>
<td>ERR_SQLSCRIPT_VAR_CYCLIC_DEPENDENCY</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2855</td>
<td>ERR_SQLSCRIPT_PARAM_NOT_INITIALIZED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2856</td>
<td>ERR_SQLSCRIPT_PARAM_MISMATCH_TABLE_TYPE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2857</td>
<td>ERR_SQLSCRIPT_CALL_OPEN_MISSING_CALL_CLOSE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2858</td>
<td>ERR_SQLSCRIPT_BUILTIN_IN_PARAM_NOTSAME_TABLE_TYPE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2859</td>
<td>ERR_SQLSCRIPT_BUILTIN_JOIN_ATTR_NOT_FOUND_IN_VAR</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2860</td>
<td>ERR_SQLSCRIPT_FUNCTION_NOT_NESTABLE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2861</td>
<td>ERR_SQLSCRIPT_CALL_CLOSE_MISSING_CALL_OPEN</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2862</td>
<td>ERR_SQLSCRIPT_TABLE_TYPE_NOT_DERIVABLE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2863</td>
<td>ERR_SQLSCRIPT_MISSING_FTC_TYPE_MAPPING</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2864</td>
<td>ERR_SQLSCRIPT_INVALID_TABLE_TYPE_NAME</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2865</td>
<td>ERR_SQLSCRIPT_DUPLICATE_ATTRIBUTE_NAME</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2866</td>
<td>ERR_SQLSCRIPT_FUNCTION_EXISTING</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2867</td>
<td>ERR_SQLSCRIPT_FUNCTION_TYPE_NOT_SUPPORTED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2868</td>
<td>ERR_SQLSCRIPT_FUNCTION_NAME_MAX_LEN</td>
<td>SqlScript</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>2869</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_ATTRIBUTE_WITH_ALIAS</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2870</td>
<td>ERR_SQLSCRIPT_INTERNAL_ERR</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2871</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_AGGREGFUN_VECTOR</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2872</td>
<td>ERR_SQLSCRIPT_FUNCTION_NAME_INVALID</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2873</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_PROJECTION_VECTOR</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2874</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_FILTER_EXPRESSION</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2875</td>
<td>ERR_SQLSCRIPT_RLANG_EXACTLY_ONE_OUTPUT_PARAM</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2876</td>
<td>ERR_SQLSCRIPT_JSLANG_EXACTLY_ONE_OUTPUT_PARAM</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2877</td>
<td>ERR_SQLSCRIPT_SQLLANG_EXACTLY_ONE_OUTPUT_PARAM</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2878</td>
<td>ERR_SQLSCRIPT GENERICLANG_EXACTLY_ONE_OUTPUT_PARAM</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2879</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_TABLE_TYPE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2880</td>
<td>ERR_SQLSCRIPT_VARIABLE_NOT_TABLE_TYPE</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2881</td>
<td>ERR_SQLSCRIPT_BUILTIN_CALCATTR_EXPRESSION_SYNTAX</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2882</td>
<td>ERR_SQLSCRIPT_BUILTIN_UNEVEN_NR_OF_PARAMS</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2883</td>
<td>ERR_SQLSCRIPT_CALCATTR_NOT_ALLOWED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2884</td>
<td>ERR_SQLSCRIPT_BUILTIN_DUPLICATE_COLUMN_NAME</td>
<td>SqlScript</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2885</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_KEY_VALUE_VECTOR</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2886</td>
<td>ERR_SQLSCRIPT_BUILTIN_CALCTR_REFERENCED_FIELD_MISSING</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2887</td>
<td>ERR_SQLSCRIPT_BUILTIN_FILTER_REFERENCED_FIELD_MISSING</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2888</td>
<td>ERR_SQLSCRIPT_BUILTIN_FILTER_EXPRESSION_SYNTAX</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2889</td>
<td>ERR_SQLSCRIPT_BUILTIN_PARAM_NOT_COL_OR_AGGR_VECTOR</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2890</td>
<td>ERR_SQLSCRIPT_TABLE_INPUT_PARAMS_NOT_SUPPORTED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>2891</td>
<td>ERR_SQLSCRIPT_TABLE_INOUT_PARAMS_NOT_SUPPORTED</td>
<td>SqlScript</td>
</tr>
<tr>
<td>3584</td>
<td>ERR_DIST_SQL</td>
<td>Distributed SQL error</td>
</tr>
<tr>
<td>3585</td>
<td>ERR_DIST_SQL_EXPR_SHIPPING_FAILURE</td>
<td>Expression shipping failure</td>
</tr>
<tr>
<td>3586</td>
<td>ERR_DIST_SQL_OPERATOR_SHIPPING_FAILURE</td>
<td>Operator shipping failure</td>
</tr>
<tr>
<td>3587</td>
<td>ERR_DIST_SQL_INVALID_PROTOCOL</td>
<td>Invalid protocol or service shutdown during distributed query execution</td>
</tr>
<tr>
<td>3588</td>
<td>ERR_DIST_SQL_SEQUENCE_SHIPPING_FAILURE</td>
<td>Sequence shipping failure</td>
</tr>
<tr>
<td>3589</td>
<td>ERR_DIST_SQL_REMOTE_EXECUTION_FAILURE</td>
<td>Remote query execution failure</td>
</tr>
<tr>
<td>3840</td>
<td>ERR_AUDITING</td>
<td>General auditing error</td>
</tr>
<tr>
<td>3841</td>
<td>ERR_AUDITING_NO_PRIV_NAME</td>
<td>Invalid privilege</td>
</tr>
<tr>
<td>3842</td>
<td>ERR_AUDITING_TRAIL_WRITER_BLOCKED</td>
<td>Audit trail writer is blocked</td>
</tr>
<tr>
<td>3843</td>
<td>ERR_AUDITING_POLICY_ALREADY_EXISTS</td>
<td>Audit policy with current name already exists</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>3844</td>
<td>ERR_AUDITING_INV_POLICY_TYPE</td>
<td>Invalid combination of audit actions</td>
</tr>
<tr>
<td>3845</td>
<td>ERR_AUDITING_INV_ACTION_STATUS</td>
<td>Invalid action status for auditing</td>
</tr>
<tr>
<td>3846</td>
<td>ERR_AUDITING_INV_LEVEL</td>
<td>Invalid auditing level</td>
</tr>
<tr>
<td>3847</td>
<td>ERR_AUDITING_INV_POLICY_NAME</td>
<td>Invalid policy name</td>
</tr>
<tr>
<td>3848</td>
<td>ERR_AUDITING_INV_ACTION_OBJECT_TYPE</td>
<td>Invalid combination of audit action and object type</td>
</tr>
<tr>
<td>3849</td>
<td>ERR_AUDITING_INV_OBJECT_TYPE</td>
<td>Audit policy for this object type not supported</td>
</tr>
<tr>
<td>3850</td>
<td>ERR_AUDITING_INV_ACTION</td>
<td>Invalid action for tenant specific audit policy</td>
</tr>
<tr>
<td>3851</td>
<td>ERR_AUDITING_INV_RETENTION</td>
<td>Number of days for retention too small or no retention for used trail types allowed at all</td>
</tr>
<tr>
<td>4096</td>
<td>ERR_PLANVIZ_GENERAL</td>
<td>[PlanViz] general error</td>
</tr>
<tr>
<td>4097</td>
<td>ERR_PLANVIZ_PIN_GENERAL</td>
<td>[PlanViz] invalid pin request</td>
</tr>
<tr>
<td>4098</td>
<td>ERR_PLANVIZ_INVALID_PLAN_GENERAL</td>
<td>[PlanViz] invalid plan</td>
</tr>
<tr>
<td>4099</td>
<td>ERR_PLANVIZ_PLAN_CACHE_GENERAL</td>
<td>[PlanViz] plan cache error</td>
</tr>
<tr>
<td>4100</td>
<td>ERR_PLANVIZ_NO_PVPARAM</td>
<td>[PlanViz] PlanVizParam does not exist</td>
</tr>
<tr>
<td>4101</td>
<td>ERR_PLANVIZ_PROC_LANG_SUPPORT</td>
<td>[PlanViz] procedure language not supported</td>
</tr>
<tr>
<td>4102</td>
<td>ERR_PLANVIZ_MARK_MIN_COST_PLAN</td>
<td>[PlanViz] cannot mark min-cost plan</td>
</tr>
<tr>
<td>4103</td>
<td>ERR_PLANVIZ_PARSE_TREE_NOT_FOUND</td>
<td>[PlanViz] parse tree not found</td>
</tr>
<tr>
<td>4104</td>
<td>ERR_PLANVIZ_PLAN_NOT_FOUND</td>
<td>[PlanViz] plan not found</td>
</tr>
<tr>
<td>4105</td>
<td>ERR_PLANVIZ_UNSUPPORTED_STMT_TYPE</td>
<td>[PlanViz] unsupported statement type</td>
</tr>
<tr>
<td>4106</td>
<td>ERR_PLANVIZ_REMOTE_EXEC_STATS</td>
<td>[PlanViz] error while gathering remote execution stats</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4107</td>
<td>ERR_PLANVIZ_ENV_NO_STATS_COLLECTOR</td>
<td>[PlanViz] exec stats collector not found</td>
</tr>
<tr>
<td>4108</td>
<td>ERR_PLANVIZ_EXPLAIN_PLAN_GENERAL</td>
<td>[PlanViz] explain plan failed</td>
</tr>
<tr>
<td>4109</td>
<td>ERR_PLANVIZ_TRACE_ONLY_GENERAL</td>
<td>[PlanViz] error in trace-only mode</td>
</tr>
<tr>
<td>4110</td>
<td>ERR_PLANVIZ_PLAN_TRACE_GENERAL</td>
<td>[PlanViz] error in Plan Trace</td>
</tr>
<tr>
<td>4161</td>
<td>ERR_REORG_GENERAL</td>
<td>Failed to execute reorganization</td>
</tr>
<tr>
<td>4162</td>
<td>ERR_REORG_TRANS_BLOCKED_GENERAL</td>
<td>Transaction blocked since runtime reorganization is in progress</td>
</tr>
<tr>
<td>4163</td>
<td>ERR_REORG_TRANS_EXISTS_GENERAL</td>
<td>Cannot start reorganization due to the transactions in execution</td>
</tr>
<tr>
<td>4192</td>
<td>ERR_LDAP</td>
<td>General LDAP error</td>
</tr>
<tr>
<td>4193</td>
<td>ERR_LDAP_CANNOT_AUTHORIZE_LOCALLY</td>
<td>Local authorization not allowed</td>
</tr>
<tr>
<td>4194</td>
<td>ERR_LDAP_CANNOT_CHANGE_AUTHORIZATION_MODE</td>
<td>Authorization mode change not allowed</td>
</tr>
<tr>
<td>4195</td>
<td>ERR_LDAP_MAPPING_ALREADY_EXISTS</td>
<td>Role to LDAP group mapping already exists</td>
</tr>
<tr>
<td>4196</td>
<td>ERR_LDAP_MAPPING_DOESNT_EXIST</td>
<td>Role to LDAP group mapping does not exist</td>
</tr>
<tr>
<td>4197</td>
<td>ERR_LDAP_PROVIDER_CREATION_FAILED</td>
<td>Creating LDAP provider failed because of internal error</td>
</tr>
<tr>
<td>4198</td>
<td>ERR_LDAP_PROVIDER_DELETION_FAILED</td>
<td>Deleting LDAP provider failed because of internal error</td>
</tr>
<tr>
<td>4199</td>
<td>ERR_LDAP_PROVIDER_ALTER_FAILED</td>
<td>Alter LDAP provider failed because of internal error</td>
</tr>
<tr>
<td>4200</td>
<td>ERR_LDAP_PROVIDER_VALIDATE_FAILED</td>
<td>Validate LDAP provider failed because of internal error</td>
</tr>
<tr>
<td>4201</td>
<td>ERR_LDAP_PROVIDER_ALREADY_EXISTS</td>
<td>LDAP provider already exists</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4202</td>
<td>ERR_LDAP_INVALID_PROVIDER_NAME</td>
<td>Invalid LDAP provider name</td>
</tr>
<tr>
<td>4203</td>
<td>ERR_LDAP_MALFORMED_CREDENTIALS</td>
<td>Credentials not provided in proper format</td>
</tr>
<tr>
<td>4204</td>
<td>ERR_LDAP_PASSWORD_MUTUAL_Exclusion_FAILED</td>
<td>Local password authentication and LDAP authentication cannot be enabled together for the same user</td>
</tr>
<tr>
<td>4225</td>
<td>ERR_PROVIDER</td>
<td>General provider error</td>
</tr>
<tr>
<td>4226</td>
<td>ERR_PROVIDER_INV_SUBJECT_NAME</td>
<td>Invalid subject name layout</td>
</tr>
<tr>
<td>4227</td>
<td>ERR_PROVIDER_INV_ISSUER_NAME</td>
<td>Invalid issuer name layout</td>
</tr>
<tr>
<td>4228</td>
<td>ERR_PROVIDER_ALREADY_EXISTS</td>
<td>Provider already exists</td>
</tr>
<tr>
<td>4229</td>
<td>ERR_PROVIDER_INVALID_PROVIDER_NAME</td>
<td>Invalid provider name</td>
</tr>
<tr>
<td>4230</td>
<td>ERR_PROVIDER_INVALID_ASSERTION</td>
<td>Invalid assertion</td>
</tr>
<tr>
<td>4231</td>
<td>ERR_PROVIDER_INVALID_MAPPED_USER_NAME</td>
<td>Invalid or empty mapped user name</td>
</tr>
<tr>
<td>4232</td>
<td>ERR_PROVIDER_ADDING_USER_MAPPING_FAILED</td>
<td>Adding a new provider user mapping failed</td>
</tr>
<tr>
<td>4233</td>
<td>ERR_PROVIDER_CREATION_FAILED</td>
<td>Creating provider failed because of internal error</td>
</tr>
<tr>
<td>4234</td>
<td>ERR_PROVIDER DELETION_FAILED</td>
<td>Deleting provider failed because of internal error</td>
</tr>
<tr>
<td>4235</td>
<td>ERR_PROVIDER ALTER_FAILED</td>
<td>Alter provider failed because of internal error</td>
</tr>
<tr>
<td>4236</td>
<td>ERR_PROVIDER DUPLICATE_ENTITY</td>
<td>EntityID already exists</td>
</tr>
<tr>
<td>4237</td>
<td>ERR_PROVIDER INVALIDENTITY</td>
<td>Invalid entity id</td>
</tr>
<tr>
<td>4238</td>
<td>ERR_PROVIDER_DUPLICATE_ISSUER</td>
<td>Duplicate provider for this issuer</td>
</tr>
<tr>
<td>4239</td>
<td>ERR_PROVIDER INVALIDCLAIM</td>
<td>Invalid claim</td>
</tr>
<tr>
<td>4240</td>
<td>ERR_PROVIDER_DUPLICATE_IDENTITY</td>
<td>Duplicate identity for this provider</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>4241</td>
<td>ERR_PROVIDER_HAS_USER_MAPPINGS</td>
<td>Provider has user mappings</td>
</tr>
<tr>
<td>4242</td>
<td>ERR_PROVIDER_DROPPING_USER_MAPPING_FAILED</td>
<td>Dropping a provider user mapping failed</td>
</tr>
<tr>
<td>4243</td>
<td>ERR_PROVIDER_DUPLICATE_SUBJECT_ISSUER</td>
<td>Duplicate provider for this subject and this issuer</td>
</tr>
<tr>
<td>4244</td>
<td>ERR_PROVIDER_INVALID_ATTRIBUTE</td>
<td>Invalid attribute</td>
</tr>
<tr>
<td>4248</td>
<td>ERR_USER_PARAMETERS</td>
<td>General user parameter error</td>
</tr>
<tr>
<td>4249</td>
<td>ERR_USER_PARAM_DUPLICATE_EMAIL_ADDRESS</td>
<td>Same email address cannot be used for different users</td>
</tr>
<tr>
<td>4250</td>
<td>ERR_USER_PARAM_PRIORITY_OUT_OF_RANGE</td>
<td>Priority out of range</td>
</tr>
<tr>
<td>4251</td>
<td>ERR_USER_PARAM_INVALID_STATEMENT_MEMORY_LIMIT</td>
<td>Invalid statement memory limit</td>
</tr>
<tr>
<td>4252</td>
<td>ERR_USER_PARAM_INVALID_STATEMENT_THREAD_LIMIT</td>
<td>Invalid statement thread limit</td>
</tr>
<tr>
<td>4253</td>
<td>ERR_USER_PARAM_INVALID_PARAMETER</td>
<td>Invalid parameter name</td>
</tr>
<tr>
<td>4273</td>
<td>ERR_KERBEROS</td>
<td>General kerberos error</td>
</tr>
<tr>
<td>4274</td>
<td>ERR_KERBEROS_DUPLICATE_PROVIDER</td>
<td>Duplicate specification of identity for KERBEROS</td>
</tr>
<tr>
<td>4275</td>
<td>ERR_KERBEROS_MISSING_PROVIDER</td>
<td>Missing specification of identity for KERBEROS</td>
</tr>
<tr>
<td>4280</td>
<td>ERR_TICKET</td>
<td>General ticket error</td>
</tr>
<tr>
<td>4281</td>
<td>ERR_TICKET_DUPLICATE</td>
<td>Duplicate specification of identity for this kind of SAP ticket</td>
</tr>
<tr>
<td>4282</td>
<td>ERR_TICKET_MISSING_PROVIDER</td>
<td>Missing specification of identity for this kind of SAP ticket</td>
</tr>
<tr>
<td>4289</td>
<td>ERR_X509</td>
<td>General X.509 error</td>
</tr>
<tr>
<td>4290</td>
<td>ERR_X509_DUPLICATE_SUBJECT_ISSUER</td>
<td>Duplicate specification of subject and issuer for X509</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4291</td>
<td>ERR_X509.Unknown_Subject_Issuer</td>
<td>Unknown specification of subject and issuer for this user</td>
</tr>
<tr>
<td>4292</td>
<td>ERR_X509.Inv_Subject_Name</td>
<td>Invalid subject name layout</td>
</tr>
<tr>
<td>4293</td>
<td>ERR_X509.Inv_Issuer_Name</td>
<td>Invalid issuer name layout</td>
</tr>
<tr>
<td>4294</td>
<td>ERR_X509.Inv_Matching_Rule</td>
<td>Invalid matching rule layout</td>
</tr>
<tr>
<td>4295</td>
<td>ERR_X509.Disabled</td>
<td>X.509 feature disabled</td>
</tr>
<tr>
<td>4296</td>
<td>ERR_X509.No_Matching_Rules</td>
<td>Provider has no matching rules</td>
</tr>
<tr>
<td>4304</td>
<td>ERR_CLIENTPKI.Unknown</td>
<td>General client PKI error</td>
</tr>
<tr>
<td>4305</td>
<td>ERR_CLIENTPKI.Enabled</td>
<td>Cannot execute when client PKI is enabled</td>
</tr>
<tr>
<td>4306</td>
<td>ERR_CLIENTPKI.Disabled</td>
<td>Cannot execute when client PKI is disabled</td>
</tr>
<tr>
<td>4307</td>
<td>ERR_CLIENTPKI.Creating_RootCA</td>
<td>Creating the client PKI root ca failed</td>
</tr>
<tr>
<td>4308</td>
<td>ERR_CLIENTPKI.Writing_RootCA_To_FS</td>
<td>Writing the client PKI root ca to the filesystem failed</td>
</tr>
<tr>
<td>4309</td>
<td>ERR_CLIENTPKI.Creating_Certificates</td>
<td>Creating the client PKI SSL certificates failed</td>
</tr>
<tr>
<td>4310</td>
<td>ERR_CLIENTPKI.Dropping_RootCA</td>
<td>Dropping client PKI root ca failed</td>
</tr>
<tr>
<td>4311</td>
<td>ERR_CLIENTPKI.Assign_Cert_to_PSE</td>
<td>Cannot assign client PKI certificate to another PSE</td>
</tr>
<tr>
<td>4320</td>
<td>ERR_SSL</td>
<td>General SSL error</td>
</tr>
<tr>
<td>4321</td>
<td>ERR_SSL.Enforce</td>
<td>Only secure connections are allowed</td>
</tr>
<tr>
<td>4336</td>
<td>ERR_USER_Remote</td>
<td>General remote user error</td>
</tr>
<tr>
<td>4337</td>
<td>ERR_USER_Remote.Exists</td>
<td>Remote user mapping already exists</td>
</tr>
<tr>
<td>4338</td>
<td>ERR_USER_Remote.Not.Exists</td>
<td>Remote user mapping does not exist</td>
</tr>
<tr>
<td>4640</td>
<td>ERR_RS_TABLE_Load_General</td>
<td>Failed to load row table</td>
</tr>
<tr>
<td>4641</td>
<td>ERR_RS_TABLE_Load_Wait_TimeOut</td>
<td>Waiting timeout for loading row table occurred</td>
</tr>
<tr>
<td>4642</td>
<td>ERR_RS_TABLE_Post_Drop</td>
<td>Failed to delete row table data</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>4672</td>
<td>ERR_DATAPROV</td>
<td>General Unified Data Provisioning error</td>
</tr>
<tr>
<td>4673</td>
<td>ERR_DATAPROV_DATA-SOURCE_DOES_NOT_EXIST</td>
<td>Data source does not exist</td>
</tr>
<tr>
<td>4674</td>
<td>ERR_DATAPROV_INVALID_LOGICAL_DATASOURCE_NAME</td>
<td>Invalid logical data source name</td>
</tr>
<tr>
<td>4675</td>
<td>ERR_DATAPROV_INVALID_DATA-FLOW_PACKAGE_NAME</td>
<td>Invalid data flow package name</td>
</tr>
<tr>
<td>4676</td>
<td>ERR_DATAPROV_INVALID_DATA-FLOW_OBJECT_NAME</td>
<td>Invalid data flow object name</td>
</tr>
<tr>
<td>4677</td>
<td>ERR_DATAPROV_DATA-FLOW_DOES_NOT_EXIST</td>
<td>Data flow does not exist</td>
</tr>
<tr>
<td>4678</td>
<td>ERR_DATAPROV_INVALID_DATAFLOW</td>
<td>Invalid data flow</td>
</tr>
<tr>
<td>4679</td>
<td>ERR_DATAPROV_INVALID_DATA-SOURCE</td>
<td>Invalid data source</td>
</tr>
<tr>
<td>4680</td>
<td>ERR_DATAPROV_COULD_NOT_GENERATE_JOB_ID</td>
<td>Could not generate job ID</td>
</tr>
<tr>
<td>4704</td>
<td>ERR_DPSERVER</td>
<td>General dpserver error occurred</td>
</tr>
<tr>
<td>4705</td>
<td>ERR_DPSERVER_SCHEMA_CHANGE</td>
<td>Schema of table changed</td>
</tr>
<tr>
<td>4706</td>
<td>ERR_DPSERVER_SCHEMA_CHANGE_ON_DATASOURCE</td>
<td>Schema of table changed in the remote source</td>
</tr>
<tr>
<td>4707</td>
<td>ERR_DPSERVER_TRUNCATE_TABLE_EVENT</td>
<td>Table truncated in the remote source</td>
</tr>
<tr>
<td>4708</td>
<td>ERR_DPSERVER_ADAPTER_FAILURE</td>
<td>Adapter failure occurred on remote source</td>
</tr>
<tr>
<td>4709</td>
<td>ERR_DPSERVER_RECEIVER_FAILURE</td>
<td>Receiver failure occurred on remote source</td>
</tr>
<tr>
<td>4710</td>
<td>ERR_DPSERVER_DISTRIBUTOR_FAILURE</td>
<td>Distributor failure occurred on remote source</td>
</tr>
<tr>
<td>4711</td>
<td>ERR_DPSERVER_ADAPTER_REMOTE_SOURCE_DOWN</td>
<td>Adapter failure</td>
</tr>
<tr>
<td>4712</td>
<td>ERR_DPSERVER_OPERATION_NOT_PERMITTED</td>
<td>Operation is not permitted</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>4713</td>
<td>ERR_DPSERVER_SCHEMA_CHANGE_</td>
<td>Source table dropped in the remote source</td>
</tr>
<tr>
<td></td>
<td>ON_DATASOURCE_DROPTABLE_OPERATION</td>
<td></td>
</tr>
<tr>
<td>4736</td>
<td>ERR_SCHEDULER_JOB_INVALID</td>
<td>Invalid scheduler job name</td>
</tr>
<tr>
<td>4737</td>
<td>ERR_SCHEDULER_JOB_EXISTS</td>
<td>Cannot use duplicate scheduler job name</td>
</tr>
<tr>
<td>4738</td>
<td>ERR_INV_USABLE_SCHEDULER_JOB</td>
<td>Invalidated scheduler job</td>
</tr>
<tr>
<td>4864</td>
<td>ERR_GEM</td>
<td>General GEM error</td>
</tr>
<tr>
<td>4865</td>
<td>ERR_GEM_WORKSPACE_NOT_EXISTS</td>
<td>GEM workspace does not exist</td>
</tr>
<tr>
<td>4866</td>
<td>ERR_GEM_WORKSPACE_SCHEMA_NOT_EXISTS</td>
<td>Schema specified for GEM workspace does not exist</td>
</tr>
<tr>
<td>4867</td>
<td>ERR_GEM_WORKSPACE_ALRE-</td>
<td>GEM workspace already exists</td>
</tr>
<tr>
<td></td>
<td>ADY_EXISTS</td>
<td></td>
</tr>
<tr>
<td>4868</td>
<td>ERR_GEM_WORKSPACE_URI_TOO_LONG</td>
<td>Workspace URI exceeds maximum allowed length</td>
</tr>
<tr>
<td>4869</td>
<td>ERR_GEM_ADD_COLUMN</td>
<td>Failed to add column</td>
</tr>
<tr>
<td>4870</td>
<td>ERR_GEM_PREP_INSERT</td>
<td>Failed in preparation to add column</td>
</tr>
<tr>
<td>4871</td>
<td>ERR_GEM_CALC</td>
<td>Calculation Scenario</td>
</tr>
<tr>
<td>4872</td>
<td>ERR_GEM_VISITOR</td>
<td>GEM Visitor</td>
</tr>
<tr>
<td>4873</td>
<td>ERR_GEM_GRAMMAR</td>
<td>GEM Grammar</td>
</tr>
<tr>
<td>4874</td>
<td>ERR_GEM_TREE_BUILDER</td>
<td>Tree Builder</td>
</tr>
<tr>
<td>4875</td>
<td>ERR_GEM_TECHTYPE_UNKNOWN</td>
<td>Techtype unknown in predicate/expression</td>
</tr>
<tr>
<td>4876</td>
<td>ERR_GEM_TECHTYPE_MISMATCH</td>
<td>Mismatch of technical types</td>
</tr>
<tr>
<td>4877</td>
<td>ERR_GEM_TERM_NOT_EXISTS</td>
<td>Term does not exist</td>
</tr>
<tr>
<td>4878</td>
<td>ERR_GEM_TERM_ALREADY_EXISTS</td>
<td>Term already exists</td>
</tr>
<tr>
<td>4879</td>
<td>ERR_GEM_VERTEX_NOT_EXISTS</td>
<td>Vertex does not exist</td>
</tr>
<tr>
<td>4880</td>
<td>ERR_GEM_VERTEX_ALREADY_EXISTS</td>
<td>Vertex already exists</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>4881</td>
<td>ERR_GEM_LOCAL_NAME_NOT_FOUND</td>
<td>Local name was not found</td>
</tr>
<tr>
<td>4882</td>
<td>ERR_GEM_LOCAL_NAME_ALREADY_EXISTS</td>
<td>Local name already exists</td>
</tr>
<tr>
<td>4883</td>
<td>ERR_GEM_UNKNOWN_FUNC</td>
<td>Unknown function</td>
</tr>
<tr>
<td>4884</td>
<td>ERR_GEM_FEATURE_NOT_SUPPORTED</td>
<td>This GEM feature is not supported</td>
</tr>
<tr>
<td>4885</td>
<td>ERR_GEM_FUNCTION</td>
<td>Error in using GEM function</td>
</tr>
<tr>
<td>4886</td>
<td>ERR_GEM_TECHTYPE_MISSING</td>
<td>Techtype not specified</td>
</tr>
<tr>
<td>4887</td>
<td>ERR_GEM_URI_MISSING</td>
<td>URI is missing</td>
</tr>
<tr>
<td>4888</td>
<td>ERR_GEM_TECHTYPE_ERROR</td>
<td>GEM technical type error</td>
</tr>
<tr>
<td>5120</td>
<td>ERR_TEXT_UNKNOWN</td>
<td>Unknown error occurred.</td>
</tr>
<tr>
<td>5121</td>
<td>ERR_TEXT_COMMON_REGISTER_OVERWRITE</td>
<td>Attempting to register an already registered object.</td>
</tr>
<tr>
<td>5122</td>
<td>ERR_TEXT_COMMON_XML_PARSER_ERROR</td>
<td>Xerces parser error.</td>
</tr>
<tr>
<td>5123</td>
<td>ERR_TEXT_COMMON_INVALID_ANALYZER</td>
<td>Invalid analyzer specified in aggregate configuration.</td>
</tr>
<tr>
<td>5124</td>
<td>ERR_TEXT_COMMON_ANALYZER_CREATE_FAILED</td>
<td>Analyzer factory create failed</td>
</tr>
<tr>
<td>5125</td>
<td>ERR_TEXT_COMMON_UNKNOWN_ANALYZER_TYPE</td>
<td>Request to create analyzer of unknown type.</td>
</tr>
<tr>
<td>5126</td>
<td>ERR_TEXT_COMMON_PIMPL_INIT_FAILED</td>
<td>Failed to create private implementation instance.</td>
</tr>
<tr>
<td>5127</td>
<td>ERR_TEXT_COMMON_NO_CONFIGURATION_FOUND</td>
<td>No configuration found.</td>
</tr>
<tr>
<td>5128</td>
<td>ERR_TEXT_COMMON_CONFIG_CREATE_FAILED</td>
<td>Failed to create configuration.</td>
</tr>
<tr>
<td>5129</td>
<td>ERR_TEXT_COMMON_LANG_DIR_READ_ERROR</td>
<td>Error reading language directory.</td>
</tr>
<tr>
<td>5130</td>
<td>ERR_TEXT_COMMON_NO_ANALYZERS_SPECIFIED</td>
<td>No analyzers were specified.</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>5131</td>
<td>ERR_TEXT_COMMON_PROPERTY_LIST_CREATE_FAILED</td>
<td>Failed to create property list.</td>
</tr>
<tr>
<td>5132</td>
<td>ERR_TEXT_COMMON_CHILD_VIEW_CREATE_FAILED</td>
<td>Failed to create child view.</td>
</tr>
<tr>
<td>5133</td>
<td>ERR_TEXT_COMMON_FAILED_BUFFER_ALLOCATION</td>
<td>Failed to allocate internal buffer.</td>
</tr>
<tr>
<td>5134</td>
<td>ERR_TEXT_COMMON_USE_OF_FOREIGN_ANNOTATION</td>
<td>Attempt to use foreign annotation.</td>
</tr>
<tr>
<td>5135</td>
<td>ERR_TEXT_COMMON_INSERT_BEFORE_INVALID</td>
<td>Attempt to insert before invalid annotation.</td>
</tr>
<tr>
<td>5136</td>
<td>ERR_TEXT_COMMON_ALLOCATION_FAILED</td>
<td>Allocation failed.</td>
</tr>
<tr>
<td>5137</td>
<td>ERR_TEXT_COMMON_INVALID_VIEW</td>
<td>Invalid view specified.</td>
</tr>
<tr>
<td>5138</td>
<td>ERR_TEXT_COMMON_INDEX_OUT_OF_RANGE</td>
<td>Index out of range.</td>
</tr>
<tr>
<td>5139</td>
<td>ERR_TEXT_COMMON_ANALYZER_BEFORE_CONFIG</td>
<td>Attempt to analyze before configuring.</td>
</tr>
<tr>
<td>5140</td>
<td>ERR_TEXT_COMMON_INVALID_ANNOTATION</td>
<td>Invalid annotation found.</td>
</tr>
<tr>
<td>5141</td>
<td>ERR_TEXT_COMMON_MISSING_REQUIRED_CONFIG</td>
<td>Configuration error: missing required configuration parameter.</td>
</tr>
<tr>
<td>5142</td>
<td>ERR_TEXT_COMMON_ANOMGR_CREATED_OUTSIDE_DOC</td>
<td>Annotation Manager created outside of document.</td>
</tr>
<tr>
<td>5143</td>
<td>ERR_TEXT_COMMON_ANQUEUE_CREATED_OUTSIDE_DOC</td>
<td>Analysis Queue created outside of document.</td>
</tr>
<tr>
<td>5144</td>
<td>ERR_TEXT_COMMON_VIEWMGR_CREATED_OUTSIDE_DOC</td>
<td>View Manager created outside of document.</td>
</tr>
<tr>
<td>5145</td>
<td>ERR_TEXT_COMMON_GRPOMGR_CREATED_OUTSIDE_DOC</td>
<td>Grouping Manager created outside of document.</td>
</tr>
<tr>
<td>5146</td>
<td>ERR_TEXT_COMMON_TOO_MANY_ANNOTATIONS_ADDED</td>
<td>Too many annotations added to annotation manager.</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5147</td>
<td>ERR_TEXT_COMMON_ARCHIVE_READ_ERROR</td>
<td>Archive read error.</td>
</tr>
<tr>
<td>5148</td>
<td>ERR_TEXT_COMMON_ARCHIVE_WRITE_ERROR</td>
<td>Archive write error.</td>
</tr>
<tr>
<td>5149</td>
<td>ERR_TEXT_COMMON_TYPE_MISMATCH_IN_SET_FIELD</td>
<td>Type mismatch when setting field value.</td>
</tr>
<tr>
<td>5150</td>
<td>ERR_TEXT_COMMON_STREAM_ALREADY_OPEN</td>
<td>Attempt to open an already open stream.</td>
</tr>
<tr>
<td>5151</td>
<td>ERR_TEXT_COMMON_STREAM_CLOSED</td>
<td>Attempt to transact with a closed stream.</td>
</tr>
<tr>
<td>5152</td>
<td>ERR_TEXT_COMMON_STREAM_READ_PAST_END_OF_STREAM</td>
<td>Attempt to read past end of stream.</td>
</tr>
<tr>
<td>5153</td>
<td>ERR_TEXT_COMMON_STREAM_FAILED_READ_OPEN</td>
<td>Failed to open stream for reading.</td>
</tr>
<tr>
<td>5154</td>
<td>ERR_TEXT_COMMON_STREAM_FAILED_WRITE_OPEN</td>
<td>Failed to open stream for writing.</td>
</tr>
<tr>
<td>5155</td>
<td>ERR_TEXT_COMMON_STREAM_READ_ERROR</td>
<td>Stream read error.</td>
</tr>
<tr>
<td>5156</td>
<td>ERR_TEXT_COMMON_INVALID_TYPE_DURING_WRITE</td>
<td>Invalid type specifier during stream write.</td>
</tr>
<tr>
<td>5157</td>
<td>ERR_TEXT_COMMON_INVALID_TYPE_DURING_READ</td>
<td>Invalid type specifier during stream read.</td>
</tr>
<tr>
<td>5158</td>
<td>ERR_TEXT_COMMON_INVALID_TYPE_DURING_PARSE</td>
<td>Attempt to initiate parsing while progressive parse is in progress.</td>
</tr>
<tr>
<td>5159</td>
<td>ERR_TEXT_COMMON_XML_PARSE_DURING_PARSE</td>
<td>Attempt to continue progressive parse before it has started.</td>
</tr>
<tr>
<td>5160</td>
<td>ERR_TEXT_COMMON_MISSING_LANGUAGE_DIR_CONFIG</td>
<td>Language directory property (LanguageDir) was not found in the Configuration.</td>
</tr>
<tr>
<td>5161</td>
<td>ERR_TEXT_COMMON_LANGUAGE_DIR_DOES_NOT_EXIST</td>
<td>Language directory specified in the configuration does not exist.</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5163</td>
<td>ERR_TEXT_COMMON_LANGUAGE_DIR_NOT_ACCESSIBLE</td>
<td>Language directory specified in the configuration is not accessible.</td>
</tr>
<tr>
<td>5164</td>
<td>ERR_TEXT_COMMON_LANGUAGE_DIR_IS_NOT_A_DIRECTORY</td>
<td>Language directory specified in the configuration is not a directory.</td>
</tr>
<tr>
<td>5165</td>
<td>ERR_TEXT_COMMON_LANGUAGE_DIR_DIRECTORY_IS_EMPTY</td>
<td>Language directory specified in the configuration is empty.</td>
</tr>
<tr>
<td>5166</td>
<td>ERR_TEXT_COMMON_LANGUAGE_DIR_COULD_NOT_BE_READ</td>
<td>Language directory specified in the configuration could not be read.</td>
</tr>
<tr>
<td>5167</td>
<td>ERR_TEXT_COMMON_LANGUAGE_DIR_NAME_IS_TOO_LONG</td>
<td>Language directory name specified in the configuration is too long.</td>
</tr>
<tr>
<td>5168</td>
<td>ERR_TEXT_COMMON_XML_IS_NOT_A_CONFIGURATOR</td>
<td>Attempt to create a Configurator from XML representation of some other type of object.</td>
</tr>
<tr>
<td>5169</td>
<td>ERR_TEXT_COMMON_INVALID_CONFIG_TYPE</td>
<td>Configuration error: invalid type for configuration parameter value.</td>
</tr>
<tr>
<td>5170</td>
<td>ERR_TEXT_COMMON_FILE_NOT_FOUND</td>
<td>File not found.</td>
</tr>
<tr>
<td>5171</td>
<td>ERR_TEXT_COMMON_LOGGERREGISTER_OVERWRITE</td>
<td>Attempted to register a MessageHandler when one was already registered. Must unregister previous one first.</td>
</tr>
<tr>
<td>5172</td>
<td>ERR_TEXT_COMMON_INVALID_LENGTH</td>
<td>Invalid length.</td>
</tr>
<tr>
<td>5173</td>
<td>ERR_TEXT_COMMON_NULL_POINTER</td>
<td>Operation attempted with a NULL pointer.</td>
</tr>
<tr>
<td>5174</td>
<td>ERR_TEXT_COMMON_INVALID_ARRAY_ARGUMENT</td>
<td>Attempt to allocate or grow an array to a size that exceeds the maximum allowed.</td>
</tr>
<tr>
<td>5175</td>
<td>ERR_TEXT_COMMON_INVALID_INPUT_BUFFER_LENGTH</td>
<td>Invalid buffer length for XML Input stream.</td>
</tr>
<tr>
<td>5176</td>
<td>ERR_TEXT_COMMON_INPUT_BUFFER_EXCEEDS_USSTRING_LIMIT_FOR_BASE64_ENCODING</td>
<td>The XML output buffer length exceeds the UString limit required for base64 encoding.</td>
</tr>
<tr>
<td>5177</td>
<td>ERR_TEXT_COMMON_EXCEEDING_OUTPUT_BUFFER_CAPACITY</td>
<td>Writing more data to the buffer would exceed the buffer capacity.</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5178</td>
<td>ERR_TEXT_COMMON_INVALID_CONFIGURATION</td>
<td>Failure when parsing configuration information.</td>
</tr>
<tr>
<td>5179</td>
<td>ERR_TEXT_COMMON_LID_INVALID_NGRAM_SIZE</td>
<td>Size of specified n-gram did not match configured size.</td>
</tr>
<tr>
<td>5180</td>
<td>ERR_TEXT_COMMON_LIB_PATH_NOT_FOUND</td>
<td>Unable to find path to TASDK shared libraries.</td>
</tr>
<tr>
<td>5181</td>
<td>ERR_TEXT_COMMON_LANGUAGE_MODEL_NOT_FOUND</td>
<td>Language model file not found for</td>
</tr>
<tr>
<td>5182</td>
<td>ERR_TEXT_COMMON_FAILED_TO_REGISTER_LIB</td>
<td>Unable to register text analysis classes in library.</td>
</tr>
<tr>
<td>5183</td>
<td>ERR_TEXT_COMMON_LANGUAGE_MODEL_NULL_POINTER</td>
<td>Cancelling XML model string parsing due to a NULL Language Model pointer</td>
</tr>
<tr>
<td>5184</td>
<td>ERR_TEXT_COMMON_LANGUAGE_MODEL_PARSER_LANGUAGE_UNMATCHED</td>
<td>The parser identified language is different from the model’s assumed language.</td>
</tr>
<tr>
<td>5185</td>
<td>ERR_TEXT_COMMON_LANGUAGE_MODEL_UNDEFINED_NGRAM_TYPE</td>
<td>Language model has an undefined Ngram type</td>
</tr>
<tr>
<td>5186</td>
<td>ERR_TEXT_COMMON_EMPTY_LANGUAGE_SET_FOR_LANGUAGE_IDENTIFICATION</td>
<td>An empty set of candidate languages was supplied for language identification.</td>
</tr>
<tr>
<td>5187</td>
<td>ERR_TEXT_COMMON_INVALID_LANGUAGE_MODEL_MANAGER</td>
<td>Invalid LanguageModel Manager.</td>
</tr>
<tr>
<td>5188</td>
<td>ERR_TEXT_COMMON_GETLOGGER_BEFORE_INIT</td>
<td>Analyzer’s getLogger() called before analyzer was initialized.</td>
</tr>
<tr>
<td>5189</td>
<td>ERR_TEXT_COMMON_INVALID_DEFAULT_LANGUAGE</td>
<td>Default language specified is not in the language list</td>
</tr>
<tr>
<td>5190</td>
<td>ERR_TEXT_COMMON_INVALID_MINIMUM_INPUT_LENGTH</td>
<td>Minimum input length has to be not less than zero.</td>
</tr>
<tr>
<td>5191</td>
<td>ERR_TEXT_COMMON_INVALID_EVALUATION_SAMPLE_SIZE</td>
<td>Evaluation sample size has to be not less than zero.</td>
</tr>
<tr>
<td>5192</td>
<td>ERR_TEXT_COMMON_INVALID_MINIMUM_CONFIDENCE</td>
<td>Minimum confidence has to be in the range [0..100]</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>5193</td>
<td>ERR_TEXT_COMMON_OVERFLOW_ERROR</td>
<td>Internal overflow error</td>
</tr>
<tr>
<td>5194</td>
<td>ERR_TEXT_COMMON_MUTEX_FAILED</td>
<td>Mutex creation failed</td>
</tr>
<tr>
<td>5195</td>
<td>ERR_TEXT_COMMON_LANGUAGE_IDENTIFICATION_MODEL_ACCESS_FAILED</td>
<td>Failed to find one or more language identification models.</td>
</tr>
<tr>
<td>5196</td>
<td>ERR_TEXT_COMMON_FILE_BASED_DICT_PATH_IN_CONFIGURATION_FILE</td>
<td>File-based dictionary path is not allowed in Text Analysis configuration file.</td>
</tr>
<tr>
<td>5197</td>
<td>ERR_TEXT_COMMON_CONVERTER_OPEN_FAILED</td>
<td>ICU Converter opening failed.</td>
</tr>
<tr>
<td>5198</td>
<td>ERR_TEXT_COMMON_CONV_TO_UNI_FAILED</td>
<td>Conversion to Unicode failed.</td>
</tr>
<tr>
<td>5199</td>
<td>ERR_TEXT_COMMON_CONV_FROM_UNI_FAILED</td>
<td>Conversion from Unicode failed.</td>
</tr>
<tr>
<td>5200</td>
<td>ERR_TEXT_COMMON_U16_APPEND_FAILED</td>
<td>UChar32 character handling failed.</td>
</tr>
<tr>
<td>5201</td>
<td>ERR_TEXT_COMMON_COMPOSED_FAILED</td>
<td>Conversion to composed form failed.</td>
</tr>
<tr>
<td>5202</td>
<td>ERR_TEXT_COMMON_DECOMPOSED_FAILED</td>
<td>Conversion to decomposed form failed.</td>
</tr>
<tr>
<td>5203</td>
<td>ERR_TEXT_COMMON_USTRING_CONSTRUCTION_FAILED</td>
<td>Constructing a UString from another type failed.</td>
</tr>
<tr>
<td>5204</td>
<td>ERR_TEXT_COMMON_CASE_CONVERSION_FAILED</td>
<td>Error converting UChar case.</td>
</tr>
<tr>
<td>5205</td>
<td>ERR_TEXT_COMMON_COMPARE_NO_CASE_FAILED</td>
<td>Case insensitive string comparison failed.</td>
</tr>
<tr>
<td>5206</td>
<td>ERR_TEXT_COMMON_FILE_BASED_RULE_SET_PATH_IN_CONFIGURATION_FILE</td>
<td>File-based rule set path is not allowed in Text Analysis configuration file.</td>
</tr>
<tr>
<td>5220</td>
<td>ERR_TEXT_EA_UNKNOWN</td>
<td>Unknown error occurred.</td>
</tr>
<tr>
<td>5221</td>
<td>ERR_TEXT_EA_NC_LOAD_ERROR</td>
<td>Error when loading dictionary.</td>
</tr>
<tr>
<td>5222</td>
<td>ERR_TEXT_EA_CONFIG_ERROR</td>
<td>Error during configuration.</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>5223</td>
<td>ERR_TEXT_EA_EXTRACT_ERROR</td>
<td>Error during extraction.</td>
</tr>
<tr>
<td>5224</td>
<td>ERR_TEXT_EA_CE_ERROR</td>
<td>Error during core extraction.</td>
</tr>
<tr>
<td>5225</td>
<td>ERR_TEXT_EA_NE_ERROR</td>
<td>Error during name extraction.</td>
</tr>
<tr>
<td>5226</td>
<td>ERR_TEXT_EA_PE_ERROR</td>
<td>Error during pattern extraction.</td>
</tr>
<tr>
<td>5227</td>
<td>ERR_TEXT_EA_PP_ERROR</td>
<td>Error during post processing.</td>
</tr>
<tr>
<td>5228</td>
<td>ERR_TEXT_EA_CR_LOAD_ERROR</td>
<td>Error when loading custom rule.</td>
</tr>
<tr>
<td>5229</td>
<td>ERR_TEXT_EA_OOB_ENTITY_LOAD</td>
<td>Error when extracting entity types.</td>
</tr>
<tr>
<td>5230</td>
<td>ERR_TEXT_EA_OOB_DIR_NOT_FOUND</td>
<td>Error when trying to access directory that does not exist.</td>
</tr>
<tr>
<td>5231</td>
<td>ERR_TEXT_EA_OOB_RES_INIT</td>
<td>Error when initializing linguistic resources.</td>
</tr>
<tr>
<td>5320</td>
<td>ERR_TEXT_LA_UNKNOWN</td>
<td>Unknown error occurred.</td>
</tr>
<tr>
<td>5321</td>
<td>ERR_TEXT_LA_INCORRECT_SCENARIO</td>
<td>Incorrect scenario requested.</td>
</tr>
<tr>
<td>5322</td>
<td>ERR_TEXT_LA_INXIGHT_SEGMENTATION</td>
<td>Error occurred during segmentation.</td>
</tr>
<tr>
<td>5323</td>
<td>ERR_TEXT_LA_SEGMENT_STRING_IFACE</td>
<td>Error occurred when creating byte interface for input string pointed to by a document segment.</td>
</tr>
<tr>
<td>5324</td>
<td>ERR_TEXT_LA_SEGMENT_GEN_IFACE</td>
<td>Error occurred when creating segmentation interface for input string pointed to by a document segment.</td>
</tr>
<tr>
<td>5325</td>
<td>ERR_TEXT_LA_SEGMENT_GEN_ENCODING</td>
<td>Error occurred when setting encoding for input string pointed to by a document segment.</td>
</tr>
<tr>
<td>5326</td>
<td>ERR_TEXT_LA_SEGMENT_GEN_LANGUAGE</td>
<td>Error occurred when setting language for input string pointed to by a document segment.</td>
</tr>
<tr>
<td>5327</td>
<td>ERR_TEXT_LA_CONTENT_STRING_IFACE</td>
<td>Error occurred when creating byte interface for document content.</td>
</tr>
<tr>
<td>5328</td>
<td>ERR_TEXT_LA_CONTENT_GEN_IFACE</td>
<td>Error occurred when creating segmentation interface for document content.</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td>5329</td>
<td>ERR_TEXT_LA_CONTENT_GEN_ENCODING</td>
<td>Error occurred when setting encoding for document content.</td>
</tr>
<tr>
<td>5330</td>
<td>ERR_TEXT_LA_CONTENT_GEN_LANGUAGE</td>
<td>Error occurred when setting language for document content.</td>
</tr>
<tr>
<td>5331</td>
<td>ERR_TEXT_LA_ANN_SEGMENT</td>
<td>Error occurred when creating a segment annotation.</td>
</tr>
<tr>
<td>5332</td>
<td>ERR_TEXT_LA_ANN_PARAGRAPH</td>
<td>Error occurred when creating a paragraph annotation.</td>
</tr>
<tr>
<td>5333</td>
<td>ERR_TEXT_LA_ANN_SENTENCE</td>
<td>Error occurred when creating a sentence annotation.</td>
</tr>
<tr>
<td>5334</td>
<td>ERR_TEXT_LA_ANN_TERM</td>
<td>Error occurred when creating a term annotation.</td>
</tr>
<tr>
<td>5335</td>
<td>ERR_TEXT_LA_SCENARIO_1</td>
<td>Error occurred when analyzing a segment.</td>
</tr>
<tr>
<td>5336</td>
<td>ERR_TEXT_LA_SCENARIO_2</td>
<td>Error occurred when analyzing a segment.</td>
</tr>
<tr>
<td>5337</td>
<td>ERR_TEXT_LA_SCENARIO_3</td>
<td>Error occurred when analyzing a segment.</td>
</tr>
<tr>
<td>5338</td>
<td>ERR_TEXT_LA_SCENARIO_4</td>
<td>Error occurred when analyzing a segment.</td>
</tr>
<tr>
<td>5339</td>
<td>ERR_TEXT_LA_SCENARIO_5</td>
<td>Error occurred when analyzing a segment.</td>
</tr>
<tr>
<td>5340</td>
<td>ERR_TEXT_LA_SCENARIO_6</td>
<td>Error occurred when analyzing a segment.</td>
</tr>
<tr>
<td>5341</td>
<td>ERR_TEXT_LA_SCENARIO_7</td>
<td>Error occurred when analyzing a segment.</td>
</tr>
<tr>
<td>5342</td>
<td>ERR_TEXT_LA_SCENARIO_8</td>
<td>Error occurred when analyzing a segment.</td>
</tr>
<tr>
<td>5343</td>
<td>ERR_TEXT_LA_LXP_RES_INIT_1</td>
<td>Error occurred when initializing LxP resource.</td>
</tr>
<tr>
<td>5344</td>
<td>ERR_TEXT_LA_LXP_RES_INIT_2</td>
<td>Error occurred when initializing LxP resource.</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5345</td>
<td>ERR_TEXT_LA_LXP_RES_INIT_3</td>
<td>Error occurred when initializing LxP resource.</td>
</tr>
<tr>
<td>5346</td>
<td>ERR_TEXT_LA_LXP_RES_INIT_4</td>
<td>Error occurred when initializing LxP resource.</td>
</tr>
<tr>
<td>5347</td>
<td>ERR_TEXT_LA_LXP_RES_INIT_5</td>
<td>Error occurred when initializing LxP resource.</td>
</tr>
<tr>
<td>5348</td>
<td>ERR_TEXT_LA_LXP_RES_INIT_6</td>
<td>Error occurred when initializing LxP resource.</td>
</tr>
<tr>
<td>5349</td>
<td>ERR_TEXT_LA_LXP_RES_INIT_7</td>
<td>Error occurred when initializing LxP resource.</td>
</tr>
<tr>
<td>5350</td>
<td>ERR_TEXT_LA_INXIGHT_TEXT_CONV</td>
<td>Error occurred during internal text manipulation.</td>
</tr>
<tr>
<td>5351</td>
<td>ERR_TEXT_LA_INXIGHT_OFFSET</td>
<td>Error occurred when getting annotation offsets.</td>
</tr>
<tr>
<td>5352</td>
<td>ERR_TEXT_LA_ADD_NORMALIZED</td>
<td>Error occurred when getting normalized case version of a token.</td>
</tr>
<tr>
<td>5353</td>
<td>ERR_TEXT_LA_WRONG_LANG_FOLDER</td>
<td>LinguisticAnalyzer was configured with nonexisting path to language resource files</td>
</tr>
<tr>
<td>5354</td>
<td>ERR_TEXT_LA_INXIGHT_FSM_INIT</td>
<td>Error occurred when initializing FSM.</td>
</tr>
<tr>
<td>5355</td>
<td>ERR_TEXT_LA_INXIGHT_FSM_QUERY</td>
<td>Error occurred when querying FSM.</td>
</tr>
<tr>
<td>5356</td>
<td>ERR_TEXT_LA_LXP_POOL_ERROR</td>
<td>Error occurred when trying to allocate the LA resource pool</td>
</tr>
<tr>
<td>5420</td>
<td>ERR_TEXT_FCA_UNKNOWN</td>
<td>Unknown error occurred in Content Filter.</td>
</tr>
<tr>
<td>5421</td>
<td>ERR_TEXT_FCA_NULL_POINTER</td>
<td>Operation attempted with a NULL pointer.</td>
</tr>
<tr>
<td>5422</td>
<td>ERR_TEXT_FCA_INVALID_ARGUMENT</td>
<td>A method was called with an invalid argument.</td>
</tr>
<tr>
<td>5423</td>
<td>ERR_TEXT_FCA_UNSUPPORTED_DOCUMENT_TYPE</td>
<td>The submitted document is of an unsupported type.</td>
</tr>
<tr>
<td>5424</td>
<td>ERR_TEXT_FCA_UNSUPPORTED_FILTER</td>
<td>The requested filter is unsupported.</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5425</td>
<td>ERR_TEXT_FCA_3RD_PARTY_FILTER_INITIALIZATION</td>
<td>Initialization of 3rd party filter failed. Verify that filter is setup correctly.</td>
</tr>
<tr>
<td>5426</td>
<td>ERR_TEXT_FCA_3RD_PARTY_FILTER_FILTERING</td>
<td>A filtering operation error occurred in the 3rd party filter.</td>
</tr>
<tr>
<td>5427</td>
<td>ERR_TEXT_FCA_FILE_OPEN</td>
<td>A file open failed.</td>
</tr>
<tr>
<td>5428</td>
<td>ERR_TEXT_FCA_FILE_READ</td>
<td>A file read failed.</td>
</tr>
<tr>
<td>5429</td>
<td>ERR_TEXT_FCA_FILE_CLOSE</td>
<td>An error occurred while closing a file.</td>
</tr>
<tr>
<td>5430</td>
<td>ERR_TEXT_FCA_FILE_WRITE</td>
<td>A file write failed.</td>
</tr>
<tr>
<td>5431</td>
<td>ERR_TEXT_FCA_UNSUPPORTED_ENCODING</td>
<td>The requested encoding is unsupported.</td>
</tr>
<tr>
<td>5432</td>
<td>ERR_TEXT_FCA_INTERNAL</td>
<td>An internal FormatConversionAnalyzer error occurred.</td>
</tr>
<tr>
<td>5433</td>
<td>ERR_TEXT_FCA_MEMORY_ALLOCATION</td>
<td>An attempt to allocate memory failed.</td>
</tr>
<tr>
<td>5434</td>
<td>ERR_TEXT_FCA_NAME_TOO_LONG</td>
<td>The name supplied for a FormatConversionAnalyzer instance is too long.</td>
</tr>
<tr>
<td>5435</td>
<td>ERR_TEXT_FCA_NOT_CONFIGURED</td>
<td>Configuration error.</td>
</tr>
<tr>
<td>5436</td>
<td>ERR_TEXT_FCA_MIME_TYPE_MISMATCH</td>
<td>Data to mime type mismatch error.</td>
</tr>
<tr>
<td>5437</td>
<td>ERR_TEXT_FCA_EXTRACT_TEXT_FROM_MEM_BUFFER</td>
<td>An error occurred while extracting text from the 3rd party filter.</td>
</tr>
<tr>
<td>5438</td>
<td>ERR_TEXT_FCA_NOT_INITIALIZED</td>
<td>Initialization error.</td>
</tr>
<tr>
<td>5439</td>
<td>ERR_TEXT_FCA_DIRECTORY_NOT_FOUND</td>
<td>Directory was not found.</td>
</tr>
<tr>
<td>5440</td>
<td>ERR_TEXT_FCA_FIELD_CREATION</td>
<td>AnnotationManager failed to create a Field object.</td>
</tr>
<tr>
<td>5441</td>
<td>ERR_TEXT_FCA_NULLCONFIGURATOR</td>
<td>The provided Configurator object is null.</td>
</tr>
<tr>
<td>5442</td>
<td>ERR_TEXT_FCA_CREATE_STREAM_FROM_MEM_BUFFER</td>
<td>An error occurred while creating 3rd party filter stream from the memory buffer.</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5443</td>
<td>ERR_TEXT_FCA_OPEN_STREAM_FROM_MEM_BUFFER</td>
<td>An error occurred while opening 3rd party filter stream created from the memory buffer.</td>
</tr>
<tr>
<td>5444</td>
<td>ERR_TEXT_FCA_SHUTDOWN</td>
<td>An error occurred while closing the 3rd party filter.</td>
</tr>
<tr>
<td>5445</td>
<td>ERR_TEXT_FCA_INVALID_OUTPUT_FORMAT</td>
<td>The requested output format is unknown.</td>
</tr>
<tr>
<td>5446</td>
<td>ERR_TEXT_FCA_INVALID_FILE_HANDLE</td>
<td>The binary data memory handle is invalid.</td>
</tr>
<tr>
<td>5447</td>
<td>ERR_TEXT_FCA_INIT_WITH_EMPTY_NAMESPACE</td>
<td>Attempted to initialize document converter with empty namespace string.</td>
</tr>
<tr>
<td>5448</td>
<td>ERR_TEXT_FCA_UNSUPPORTED_FORMAT</td>
<td>The input is in an unsupported format.</td>
</tr>
<tr>
<td>5449</td>
<td>ERR_TEXT_FCA_FORMAT_WITH_NO_TEXT</td>
<td>The input contains no text for Text Analysis to process.</td>
</tr>
<tr>
<td>5450</td>
<td>ERR_TEXT_FCA_UNDEFINED</td>
<td>Undefined error code.</td>
</tr>
<tr>
<td>5520</td>
<td>ERR_TEXT_SA_UNKNOWN</td>
<td>Unknown error occurred in Structure Analyzer.</td>
</tr>
<tr>
<td>5521</td>
<td>ERR_TEXT_SA_LA_CALL</td>
<td>Lxplatform call failed.</td>
</tr>
<tr>
<td>5522</td>
<td>ERR_TEXT_SA_TEXT_CONFIGURE</td>
<td>Failed to configure TextNormalizer.</td>
</tr>
<tr>
<td>5523</td>
<td>ERR_TEXT_SA_TEXT_PROCESS</td>
<td>Failed to process TextNormalizer.</td>
</tr>
<tr>
<td>5524</td>
<td>ERR_TEXT_SA_HTML_CONFIGURE</td>
<td>Failed to configure HTMLDetagger.</td>
</tr>
<tr>
<td>5525</td>
<td>ERR_TEXT_SA_HTML_PROCESS</td>
<td>Failed to process HTMLDetagger.</td>
</tr>
<tr>
<td>5526</td>
<td>ERR_TEXT_SA_XML_CONFIGURE</td>
<td>Failed to configure XMLDetagger.</td>
</tr>
<tr>
<td>5527</td>
<td>ERR_TEXT_SA_XML_PROCESS</td>
<td>Failed to process XMLDetagger.</td>
</tr>
<tr>
<td>5528</td>
<td>ERR_TEXT_SA_LANGUAGE_CONFIGURE</td>
<td>Failed to configure LanguageDetector.</td>
</tr>
<tr>
<td>5529</td>
<td>ERR_TEXT_SA_LANGUAGE_PROCESS</td>
<td>Failed to process LanguageDetector.</td>
</tr>
<tr>
<td>5530</td>
<td>ERR_TEXT_SA_INVALID_TEXT_LIMIT</td>
<td>Invalid text limit value.</td>
</tr>
<tr>
<td>5531</td>
<td>ERR_TEXT_SA_LANGUAGE_MODEL_NOT_LOADED</td>
<td>Requested language not loaded.</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>5533</td>
<td>ERR_TEXT_PREPROCESSOR_FAILED</td>
<td>Preprocessor: failed</td>
</tr>
<tr>
<td>5534</td>
<td>ERR_TEXT_PREPROCESSOR_INCONSISTENT_DATA</td>
<td>Preprocessor: inconsistent data</td>
</tr>
<tr>
<td>5535</td>
<td>ERR_TEXT_PREPROCESSOR_EXCEPTION</td>
<td>Preprocessor: exception</td>
</tr>
<tr>
<td>5536</td>
<td>ERR_TEXT_PREPROCESSOR_INVALID_INPUT_DOC</td>
<td>Preprocessor: invalid input document</td>
</tr>
<tr>
<td>5537</td>
<td>ERR_TEXT_PREPROCESSOR_INVALID_OUTPUT_DOC</td>
<td>Preprocessor: invalid output document</td>
</tr>
<tr>
<td>5538</td>
<td>ERR_TEXT_PREPROCESSOR_DOC_NOT_CHANGED</td>
<td>Preprocessor: doc not changed</td>
</tr>
<tr>
<td>5539</td>
<td>ERR_TEXT_PREPROCESSOR_FILTER_ERROR</td>
<td>Preprocessor: filter error</td>
</tr>
<tr>
<td>5540</td>
<td>ERR_TEXT_PREPROCESSOR_CLIENT_NOT_REACHABLE</td>
<td>PreprocessorClient: Communication error</td>
</tr>
<tr>
<td>5541</td>
<td>ERR_TEXT_PREPROCESSOR_ACTIVITY_ERROR</td>
<td>Preprocessor: Error in Activity</td>
</tr>
<tr>
<td>5542</td>
<td>ERR_TEXT_PREPROCESSOR_COMMUNICATION_ERROR</td>
<td>Preprocessor: Communication error</td>
</tr>
<tr>
<td>5543</td>
<td>ERR_TEXT_PREPROCESSOR_PROTOCOL_MISMATCH</td>
<td>Preprocessor: protocol mismatch</td>
</tr>
<tr>
<td>5544</td>
<td>ERR_TEXT_PREPROCESSOR_REQUEST_CANCELED</td>
<td>Preprocessor: request canceled</td>
</tr>
<tr>
<td>5547</td>
<td>ERR_TEXT_LEXICON_ERROR</td>
<td>Lexicon: general failure</td>
</tr>
<tr>
<td>5549</td>
<td>ERR_TEXT_DAF_TA_RESULT_INCONSISTENT</td>
<td>Inconsistent TAResult</td>
</tr>
<tr>
<td>5550</td>
<td>ERR_TEXT_DAF_INVALID</td>
<td>Invalid DAF</td>
</tr>
<tr>
<td>5633</td>
<td>ERR_CERTADM_UNKNOWN</td>
<td>Unknown certificate administration error occurred</td>
</tr>
<tr>
<td>5634</td>
<td>ERR_CERTADM_INVALID_CERT_DEFINITION</td>
<td>Certificate definition inconsistent</td>
</tr>
<tr>
<td>5635</td>
<td>ERR_CERTADM_EXIST_CERTIFICATE</td>
<td>Certificate already exists</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5636</td>
<td>ERR_CERTADM_NON_EXIST_CERTIFICATE</td>
<td>Certificate could not be found in certificate store</td>
</tr>
<tr>
<td>5637</td>
<td>ERR_CERTADM_CERTIFICATE_IN_USE</td>
<td>Certificate could not be dropped because it is still in use by at least one PSE</td>
</tr>
<tr>
<td>5638</td>
<td>ERR_CERTADM_EXIST_PSE</td>
<td>PSE with same name already exists</td>
</tr>
<tr>
<td>5639</td>
<td>ERR_CERTADM_NON_EXIST_PSE</td>
<td>PSE could not be found</td>
</tr>
<tr>
<td>5640</td>
<td>ERR_CERTADM_INVALID_PURPOSE</td>
<td>Invalid purpose for PSE</td>
</tr>
<tr>
<td>5641</td>
<td>ERR_CERTADM_NO_SUCH_CERTIFICATE_IN_PSE</td>
<td>PSE does not contain such certificate</td>
</tr>
<tr>
<td>5642</td>
<td>ERR_CERTADM_PSE_WITH_DIFFERENT_PURPOSE</td>
<td>PSE is already set to a different purpose</td>
</tr>
<tr>
<td>5643</td>
<td>ERR_CERTADM_INVALID_PRIVATE_KEY</td>
<td>Private key provided with own certificate is missing or invalid</td>
</tr>
<tr>
<td>5644</td>
<td>ERR_CERTADM_NO_CERT_For_PRIVATE_KEY</td>
<td>No certificate found for provided key</td>
</tr>
<tr>
<td>5645</td>
<td>ERR_CERTADM_INCOMPLETE_CHAIN</td>
<td>Incomplete certificate chain</td>
</tr>
<tr>
<td>5646</td>
<td>ERR_CERTADM_INVALID_CHAIN</td>
<td>Invalid certificate chain</td>
</tr>
<tr>
<td>5647</td>
<td>ERR_CERTADM_DANGLING_CERTIFICATES</td>
<td>Invalid certificate chain: dangling certificates in PEM</td>
</tr>
<tr>
<td>5648</td>
<td>ERR_CERTADM_PSE_WITH_NO_PURPOSE</td>
<td>PSE is not set to any purpose</td>
</tr>
<tr>
<td>5649</td>
<td>ERR_CERTADM_OBJECT_NOT_FOUND</td>
<td>Purpose object not found</td>
</tr>
<tr>
<td>5650</td>
<td>ERR_CERTADM_OBJECT_ALREADY_IN_USE</td>
<td>Purpose object already in use by another PSE</td>
</tr>
<tr>
<td>5651</td>
<td>ERR_CERTADM_INVALID_HOSTNAME</td>
<td>Invalid hostname format</td>
</tr>
<tr>
<td>5652</td>
<td>ERR_CERTADM_PSE_WITHOUT_OTHER_PURPOSE_OBJECT</td>
<td>PSE has no other purpose object</td>
</tr>
<tr>
<td>5653</td>
<td>ERR_CERTADM_PSE_WITHOUT_OTHER_HOST</td>
<td>PSE has no other host</td>
</tr>
<tr>
<td>5654</td>
<td>ERR_CERTADM_CERTIFICATE_EXPIRED</td>
<td>Certificate is already expired</td>
</tr>
<tr>
<td>Code</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5655</td>
<td>ERR_CERTADM_CHAIN_CERTIFICATE_EXPIRED</td>
<td>Invalid certificate chain: contains expired certificates</td>
</tr>
<tr>
<td>5656</td>
<td>ERR_CERTADM_UNSET_OWN_WITH_PURPOSE</td>
<td>Can't unset own certificate if the PSE still has the purpose SSL</td>
</tr>
<tr>
<td>5657</td>
<td>ERR_CERTADM_PSE_PATH_PURPOSE_BLOCKED_BY_CONFIG</td>
<td>PSE purpose blocked by configuration</td>
</tr>
<tr>
<td>5658</td>
<td>ERR_CERTADM_INVALID_PUBKEY_DEFINITION</td>
<td>Public key definition invalid or not supported</td>
</tr>
<tr>
<td>5659</td>
<td>ERR_CERTADM_EXST_PUBKEY</td>
<td>Public key with same name already exists</td>
</tr>
<tr>
<td>5660</td>
<td>ERR_CERTADM_NON_EXST_PUBKEY</td>
<td>Public key could not be found</td>
</tr>
<tr>
<td>5661</td>
<td>ERR_CERTADM_PUBKEY_IN_USE</td>
<td>Public key could not be dropped because it is still in use by at least one PSE</td>
</tr>
<tr>
<td>5662</td>
<td>ERR_CERTADM_NO_SUCH_PUBKEY_IN_PSE</td>
<td>PSE does not contain such public key</td>
</tr>
<tr>
<td>5663</td>
<td>ERR_CERTADM_RELOAD_PSE_FAILURE</td>
<td>Error reloading file-based PSEs</td>
</tr>
<tr>
<td>5712</td>
<td>ERR_PFCG_INVALID_CONFIGURATION</td>
<td>Pfcg: invalid configuration</td>
</tr>
<tr>
<td>5734</td>
<td>ERR_MASKING_UNKNOWN</td>
<td>Masking: unknown error occurred</td>
</tr>
<tr>
<td>5735</td>
<td>ERR_MASKING_INVALID_MASK_EXPRESSION</td>
<td>Masking: invalid mask expression</td>
</tr>
<tr>
<td>5736</td>
<td>ERR_MASKING_INVALID_COLUMN_DATATYPE</td>
<td>Masking: not supported data type</td>
</tr>
<tr>
<td>5737</td>
<td>ERR_MASKING_INVALID_MASK_COLUMN</td>
<td>Masking: invalid mask column</td>
</tr>
<tr>
<td>0</td>
<td>ERR_LTT_NO_ERROR</td>
<td>Default constructed empty exception object</td>
</tr>
</tbody>
</table>

**Related Information**

M_ERROR_CODES System View [page 1883]
SQL Error Reference (SAP HANA Troubleshooting Guide)
### 4.13 ANSI SQL Compliance

Information about SAP HANA’s compliance with the mandatory features of the ANSI SQL/2016 standard.

#### 4.13.1 Compliance With Mandatory ANSI SQL Features

Compliance with mandatory ANSI SQL features by feature ID.

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
<th>Support</th>
<th>Description</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>E011</td>
<td>Numeric data types</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E011-01 INTEGER and SMALL-INT data types (including all spellings)</td>
<td>Full Support</td>
<td></td>
<td>Numeric Data Types [page 48]</td>
</tr>
<tr>
<td></td>
<td>E011-02 REAL, DOUBLE PRECISION, and FLOAT data types</td>
<td>Full Support</td>
<td></td>
<td>Numeric Data Types [page 48]</td>
</tr>
<tr>
<td></td>
<td>E011-03 DECIMAL and NUMERIC data types</td>
<td>Full Support</td>
<td></td>
<td>Numeric Data Types [page 48]</td>
</tr>
<tr>
<td></td>
<td>E011-04 Arithmetic operators</td>
<td>Full Support</td>
<td></td>
<td>Operators [page 51]</td>
</tr>
<tr>
<td></td>
<td>E011-05 Numeric comparison</td>
<td>Full Support</td>
<td></td>
<td>Operators [page 51]</td>
</tr>
<tr>
<td></td>
<td>E011-06 Implicit casting among the numeric data types</td>
<td>Full Support</td>
<td></td>
<td>Data Type Conversion [page 37]</td>
</tr>
<tr>
<td>E021</td>
<td>Character string types</td>
<td>Partial Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E021-01 CHARACTER data type</td>
<td>No Support</td>
<td>Fixed length character is not supported</td>
<td>Character String Data Types [page 36]</td>
</tr>
<tr>
<td></td>
<td>E021-02 CHARACTER VARYING data type</td>
<td>Full Support</td>
<td></td>
<td>Character String Data Types [page 36]</td>
</tr>
<tr>
<td></td>
<td>E021-03 Character literals</td>
<td>Full Support</td>
<td></td>
<td>Null and zero length are distinguished</td>
</tr>
<tr>
<td></td>
<td>E021-04 CHARACTER_LENGTH function</td>
<td>Equivalent Support</td>
<td></td>
<td>LENGTH Function (String) [page 243]</td>
</tr>
<tr>
<td></td>
<td>E021-05 OCTET_LENGTH function</td>
<td>No Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E021-06 SUBSTRING function</td>
<td>Equivalent Support</td>
<td>SUBSTRING is supported with some syntax differences.</td>
<td>SUBSTRING Function (String) [page 359]</td>
</tr>
<tr>
<td></td>
<td>E021-07 Character concatenation</td>
<td>Full Support</td>
<td>Both CONCAT and</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Feature</td>
<td>Support</td>
<td>Description</td>
<td>Links</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>E021-08</td>
<td>UPPER and LOWER functions</td>
<td>Full Support</td>
<td></td>
<td>UPPER Function (String) [page 389]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LOWER Function (String) [page 255]</td>
</tr>
<tr>
<td>E021-09</td>
<td>TRIM function</td>
<td>Full Support</td>
<td></td>
<td>TRIM Function (String) [page 385]</td>
</tr>
<tr>
<td>E021-10</td>
<td>Implicit casting among the fixed-length and variable-length character string types</td>
<td>No Support</td>
<td>Fixed-length character is not supported. Therefore, comparison or casting against the different type is not supported.</td>
<td></td>
</tr>
<tr>
<td>E021-11</td>
<td>POSITION function</td>
<td>Partial Support</td>
<td>The LOCATE function is not supported for character and binary positions.</td>
<td>LOCATE Function (String) [page 250]</td>
</tr>
<tr>
<td>E021-12</td>
<td>Character comparison</td>
<td>Partial Support</td>
<td>Fixed-length character is not supported.</td>
<td></td>
</tr>
<tr>
<td>E031</td>
<td>Identifiers</td>
<td>Full Support</td>
<td>&quot;Special characters&quot; are allowed in identifier names.</td>
<td></td>
</tr>
<tr>
<td>E031-01</td>
<td>Delimited identifiers</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E031-02</td>
<td>Lower case identifiers</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E031-03</td>
<td>Trailing underscore</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E051</td>
<td>Basic query specification</td>
<td>Partial Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E051-01</td>
<td>SELECT DISTINCT</td>
<td>Full Support</td>
<td></td>
<td>SELECT Statement (Data Manipulation) [page 1104]</td>
</tr>
<tr>
<td>E051-02</td>
<td>GROUP BY clause</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E051-04</td>
<td>GROUP BY can contain columns not in the select list</td>
<td>Full Support</td>
<td>For example, SELECT COUNT(&lt;table_name&gt;) FROM tables GROUP BY &lt;schema_name&gt;</td>
<td></td>
</tr>
<tr>
<td>E051-05</td>
<td>Select list items can be renamed</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E051-06</td>
<td>HAVING clause</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E051-07</td>
<td>Qualified * in select list</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E051-08</td>
<td>Correlation names in the FROM clause</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Feature</td>
<td>Support</td>
<td>Description</td>
<td>Links</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>E051-09</td>
<td>Rename columns in the FROM clause</td>
<td>Partial</td>
<td>Supported by a derived table or subquery.</td>
<td></td>
</tr>
<tr>
<td>E061-01</td>
<td>Comparison predicate</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E061-02</td>
<td>BETWEEN predicate</td>
<td>Full Support</td>
<td></td>
<td>BETWEEN Predicate [page 63]</td>
</tr>
<tr>
<td>E061-03</td>
<td>IN predicate with list of values</td>
<td>Full Support</td>
<td></td>
<td>IN Predicate [page 69]</td>
</tr>
<tr>
<td>E061-04</td>
<td>LIKE predicate</td>
<td>Full Support</td>
<td></td>
<td>LIKE Predicate [page 71]</td>
</tr>
<tr>
<td>E061-05</td>
<td>LIKE predicate: ESCAPE clause</td>
<td>Full Support</td>
<td></td>
<td>LIKE Predicate [page 71]</td>
</tr>
<tr>
<td>E061-06</td>
<td>NULL predicate</td>
<td>Full Support</td>
<td></td>
<td>NULL Predicate [page 74]</td>
</tr>
<tr>
<td>E061-07</td>
<td>Quantified comparison predicate</td>
<td>Full Support</td>
<td></td>
<td>Comparison Predicates (ANY, SOME, and ALL) [page 62]</td>
</tr>
<tr>
<td>E061-08</td>
<td>EXISTS predicate</td>
<td>Full Support</td>
<td></td>
<td>EXISTS Predicate [page 68]</td>
</tr>
<tr>
<td>E061-09</td>
<td>Subqueries in comparison predicate</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E061-11</td>
<td>Subqueries in IN predicate</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E061-12</td>
<td>Subqueries in quantified comparison predicate</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E061-13</td>
<td>Correlated subqueries</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E061-14</td>
<td>Search condition</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E071-01</td>
<td>Basic query expressions</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E071-02</td>
<td>UNION DISTINCT table operator</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E071-03</td>
<td>UNION ALL table operator</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E071-04</td>
<td>EXCEPT DISTINCT table operator</td>
<td>Full Support</td>
<td></td>
<td>EXCEPT is EXCEPT DISTINCT</td>
</tr>
<tr>
<td>E071-05</td>
<td>Columns combined via table operators need not have exactly the same data type.</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Feature</td>
<td>Support</td>
<td>Description</td>
<td>Links</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>E071-06</td>
<td>Table operators in sub-queries</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E081</td>
<td>Basic Privileges</td>
<td>Full Support</td>
<td></td>
<td>GRANT Statement (Access Control) [page 1010]</td>
</tr>
<tr>
<td>E081-01</td>
<td>SELECT privilege at the table level</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E081-02</td>
<td>DELETE privilege</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E081-03</td>
<td>INSERT privilege at the table level</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E081-04</td>
<td>UPDATE privilege at the table level</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E081-05</td>
<td>UPDATE privilege at the column level</td>
<td>No Support</td>
<td>Column level privilege is currently not supported.</td>
<td></td>
</tr>
<tr>
<td>E081-06</td>
<td>REFERENCES privilege at the table level</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E081-07</td>
<td>REFERENCES privilege at the column level</td>
<td>No Support</td>
<td>Column level privilege is currently not supported.</td>
<td></td>
</tr>
<tr>
<td>E081-08</td>
<td>WITH GRANT OPTION</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E081-09</td>
<td>USAGE privilege</td>
<td>Partial Support</td>
<td>REFERENCES instead of USAGE for schema and tables is used</td>
<td></td>
</tr>
<tr>
<td>E081-10</td>
<td>EXECUTE privilege</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E091</td>
<td>Set functions</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E091-01</td>
<td>AVG</td>
<td>Full Support</td>
<td></td>
<td>AVG Function (Aggregate) [page 110]</td>
</tr>
<tr>
<td>E091-02</td>
<td>COUNT</td>
<td>Full Support</td>
<td></td>
<td>COUNT Function (Aggregate) [page 155]</td>
</tr>
<tr>
<td>E091-03</td>
<td>MAX</td>
<td>Full Support</td>
<td></td>
<td>MAX Function (Aggregate) [page 259]</td>
</tr>
<tr>
<td>E091-04</td>
<td>MIN</td>
<td>Full Support</td>
<td></td>
<td>MIN Function (Aggregate) [page 266]</td>
</tr>
<tr>
<td>E091-05</td>
<td>SUM</td>
<td>Full Support</td>
<td></td>
<td>SUM Function (Aggregate) [page 360]</td>
</tr>
<tr>
<td>E091-06</td>
<td>ALL quantifier</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E091-07</td>
<td>DISTINCT quantifier</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E101</td>
<td>Basic data manipulation</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Feature</td>
<td>Support</td>
<td>Description</td>
<td>Links</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>E101-01</td>
<td>INSERT statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E101-03</td>
<td>Searched UPDATE statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E101-04</td>
<td>Searched DELETE statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E111</td>
<td>Single row SELECT statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E121</td>
<td>Basic cursor support</td>
<td>Partial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E121-01</td>
<td>DECLARE CURSOR</td>
<td>Partial</td>
<td>FOR READ ONLY is not supported.</td>
<td></td>
</tr>
<tr>
<td>E121-02</td>
<td>ORDER BY columns</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E121-03</td>
<td>Value expressions in ORDER BY clause</td>
<td>Full Support</td>
<td>SELECT (id2-id) AS test, city FROM table4 ORDER BY id2 - id;</td>
<td></td>
</tr>
<tr>
<td>E121-04</td>
<td>OPEN statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E121-06</td>
<td>Positioned UPDATE statement</td>
<td>Full Support</td>
<td>Updateable cursor supported from HANA 2 SPS03</td>
<td></td>
</tr>
<tr>
<td>E121-07</td>
<td>Positioned DELETE statement</td>
<td>Full Support</td>
<td>Updateable cursor supported from HANA 2 SPS03</td>
<td></td>
</tr>
<tr>
<td>E121-08</td>
<td>CLOSE statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E121-10</td>
<td>FETCH statement: implicit NEXT</td>
<td>Partial</td>
<td>Fetch orientation is not supported. Always implicit NEXT.</td>
<td></td>
</tr>
<tr>
<td>E121-17</td>
<td>WITH HOLD cursors</td>
<td>Full Support</td>
<td>Cursor on HOLD supported from HANA 2 SPS04</td>
<td></td>
</tr>
<tr>
<td>E131</td>
<td>Null value support (nulls in lieu of values)</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E141</td>
<td>Basic integrity constraints</td>
<td>Partial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E141-01</td>
<td>NOT NULL constraints</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E141-02</td>
<td>UNIQUE constraints of NOT NULL columns</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E141-03</td>
<td>PRIMARY KEY constraints</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Feature</td>
<td>Support</td>
<td>Description</td>
<td>Links</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>E141-04</td>
<td>Basic FOREIGN KEY constraint with the NO ACTION default for both referential delete action and referential update action.</td>
<td>Partial Support</td>
<td>Support as RESTRICT mode as default</td>
<td></td>
</tr>
<tr>
<td>E141-06</td>
<td>CHECK constraints</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E141-07</td>
<td>Column defaults</td>
<td>Partial Support</td>
<td>Does not include USER, CURRENT_ROLE, SYSTEM_USER, CURRENT_CATALOG, CURRENT_PATH</td>
<td></td>
</tr>
<tr>
<td>E141-08</td>
<td>NOT NULL inferred on PRIMARY KEY</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E141-10</td>
<td>Names in a foreign key can be specified in any order</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E151</td>
<td>Transaction support</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E151-01</td>
<td>COMMIT statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E151-02</td>
<td>ROLLBACK statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E152</td>
<td>Basic SET TRANSACTION statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E152-01</td>
<td>SET TRANSACTION statement: ISOLATION LEVEL SERIALIZABLE clause</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E152-02</td>
<td>SET TRANSACTION statement: READ ONLY and READ WRITE clauses</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E153</td>
<td>Updatable queries with subqueries</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E161</td>
<td>SQL comments using leading double minus</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E171</td>
<td>SQLSTATE support</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E182</td>
<td>Module language</td>
<td>Partial Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F021</td>
<td>Basic information schema</td>
<td>Equivalent Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F021-01</td>
<td>COLUMNS view</td>
<td>Equivalent Support</td>
<td>Supported with different views.</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Feature</td>
<td>Support</td>
<td>Description</td>
<td>Links</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>F021-02</td>
<td>TABLES view</td>
<td>Equivalent</td>
<td>Supported with different views.</td>
<td></td>
</tr>
<tr>
<td>F021-03</td>
<td>VIEWS view</td>
<td>Equivalent</td>
<td>Supported with different views.</td>
<td></td>
</tr>
<tr>
<td>F021-04</td>
<td>TABLE_CONSTRAINTS view</td>
<td>Equivalent</td>
<td>Supported with different views.</td>
<td></td>
</tr>
<tr>
<td>F021-05</td>
<td>REFERENTIAL_CONSTRAINTS view</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F021-06</td>
<td>CHECK_CONSTRAINTS view</td>
<td>Equivalent</td>
<td>Supported with different views.</td>
<td></td>
</tr>
<tr>
<td>F021-07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F031</td>
<td>Basic schema manipulation</td>
<td>Partial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F031-01</td>
<td>CREATE TABLE statement to create persistent base tables</td>
<td>Full Support</td>
<td>CREATE TABLE Statement (Data Definition) [page 825]</td>
<td></td>
</tr>
<tr>
<td>F031-02</td>
<td>CREATE VIEW statement</td>
<td>Full Support</td>
<td>CREATE VIEW Statement (Data Definition) [page 904]</td>
<td></td>
</tr>
<tr>
<td>F031-03</td>
<td>GRANT statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F031-04</td>
<td>ALTER TABLE statement: ADD COLUMN clause</td>
<td>Equivalent Support</td>
<td>ALTER TABLE Statement (Data Definition) [page 589]</td>
<td>The optional keyword COLUMN in this syntax is not supported. Also, it requires that the column definition be enclosed in parentheses, unlike the standard.</td>
</tr>
<tr>
<td>F031-13</td>
<td>DROP TABLE statement: RESTRICT clause</td>
<td>Full Support</td>
<td>DROP TABLE Statement (Data Definition) [page 980]</td>
<td></td>
</tr>
<tr>
<td>F031-16</td>
<td>DROP VIEW statement: RESTRICT clause</td>
<td>Full Support</td>
<td>DROP VIEW Statement (Data Definition) [page 988]</td>
<td></td>
</tr>
<tr>
<td>F031-19</td>
<td>REVOKE statement: RESTRICT clause</td>
<td>No Support</td>
<td>REVOKE Statement (Access Control) [page 1098]</td>
<td></td>
</tr>
<tr>
<td>F041</td>
<td>Basic joined table</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F041-01</td>
<td>Inner join (but not necessarily the INNER keyword)</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F041-02</td>
<td>INNER keyword</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F041-03</td>
<td>LEFT OUTER JOIN</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F041-04</td>
<td>RIGHT OUTER JOIN</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Feature</td>
<td>Support</td>
<td>Description</td>
<td>Links</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>F041-05</td>
<td>Outer joins can be nested</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F041-07</td>
<td>The inner table in a left or right outer join can also be used in an inner join</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F041-08</td>
<td>All comparison operators are supported (rather than just =)</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F051</td>
<td>Basic date and time</td>
<td>Equivalent Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F051-01</td>
<td>DATE data type (including support of DATE literal)</td>
<td>Full Support</td>
<td>DATE is a synonym for DAYDATE</td>
<td>Datetime Data Types [page 40]</td>
</tr>
<tr>
<td>F051-02</td>
<td>TIME data type (including support of TIME literal) with fractional seconds precision of at least 0.</td>
<td>Full Support</td>
<td></td>
<td>Datetime Data Types [page 40]</td>
</tr>
<tr>
<td>F051-03</td>
<td>TIMESTAMP data type (including support of TIMES-TAMP literal) with fractional seconds precision of at least 0 and 6.</td>
<td>Full Support</td>
<td></td>
<td>Datetime Data Types [page 40]</td>
</tr>
<tr>
<td>F051-04</td>
<td>Comparison predicate on DATE, TIME, and TIMESTAMP data types</td>
<td>Full Support</td>
<td>DATE is a synonym for DAYDATE</td>
<td>Datetime Data Types [page 40]</td>
</tr>
<tr>
<td>F051-05</td>
<td>Explicit CAST between date- time types and character string types</td>
<td>Full Support</td>
<td></td>
<td>CAST Function (Data Type Conversion) [page 125]</td>
</tr>
<tr>
<td>F051-06</td>
<td>CURRENT_DATE</td>
<td>Equivalent Support</td>
<td>SYSDATE can be used. UTCDATE, UTCTIME, UTCTIMESTAMP can be used instead.</td>
<td>CURRENT_DATE Function (Date-time) [page 167]</td>
</tr>
<tr>
<td>F051-07</td>
<td>LOCALTIME</td>
<td>Equivalent Support</td>
<td>LOCALTIME = CURRENT_TIME without timezone</td>
<td>CURRENT_DATE Function (Date-time) [page 167]</td>
</tr>
<tr>
<td>F051-08</td>
<td>LOCALTIMESTAMP</td>
<td>Equivalent Support</td>
<td>LOCALTIMESTAMP = CURRENT_TIMESTAMP without timezone</td>
<td>CURRENT_DATE Function (Date-time) [page 167] UCTOLOCAL Function (Datet ime) [page 390]</td>
</tr>
<tr>
<td>F081</td>
<td>UNION and EXCEPT in views</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F131</td>
<td>Grouped operations</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Feature</td>
<td>Support</td>
<td>Description</td>
<td>Links</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>F131-01</td>
<td>WHERE, GROUP BY, and HAVING clauses supported in queries with grouped views</td>
<td>Full Support</td>
<td></td>
<td>SELECT Statement (Data Manipulation) [page 1104]</td>
</tr>
<tr>
<td>F131-02</td>
<td>Multiple tables supported in queries with grouped views</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F131-03</td>
<td>Set functions supported in queries with grouped views</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F131-04</td>
<td>Subqueries with GROUP BY and HAVING clauses and grouped views</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F131-05</td>
<td>Single row SELECT with GROUP BY and HAVING clauses and grouped views</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F181</td>
<td>Multiple module support</td>
<td>No Support</td>
<td>NOTE 510 — The ability to associate multiple host compilation units with a single SQL session at one time.</td>
<td></td>
</tr>
<tr>
<td>F201</td>
<td>CAST function</td>
<td>Full Support</td>
<td></td>
<td>CAST Function (Data Type Conversion) [page 125]</td>
</tr>
<tr>
<td>F221</td>
<td>Explicit defaults</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F261</td>
<td>CASE expression</td>
<td>Full Support</td>
<td></td>
<td>Expressions [page 56]</td>
</tr>
<tr>
<td>F261-01</td>
<td>Simple CASE</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F261-02</td>
<td>Searched CASE</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F261-03</td>
<td>NULLIF</td>
<td>Full Support</td>
<td></td>
<td>NULLIF Function (Miscellaneous) [page 282]</td>
</tr>
<tr>
<td>F261-04</td>
<td>COALESCE</td>
<td>Full Support</td>
<td></td>
<td>COALESCE Function (Miscellaneous) [page 128]</td>
</tr>
<tr>
<td>F311</td>
<td>Schema definition statement</td>
<td>Partial Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F311-01</td>
<td>CREATE SCHEMA</td>
<td>Full Support</td>
<td>Only users with “CREATE SCHEMA” privilege can create schemas.</td>
<td>CREATE SCHEMA Statement (Data Definition) [page 807]</td>
</tr>
<tr>
<td>F311-02</td>
<td>CREATE TABLE for persistent base tables</td>
<td>Partial Support</td>
<td>CREATE TABLE is supported as a separate statement from CREATE SCHEMA</td>
<td>CREATE TABLE Statement (Data Definition) [page 825]</td>
</tr>
<tr>
<td>ID</td>
<td>Feature</td>
<td>Support</td>
<td>Description</td>
<td>Links</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>F311-03</td>
<td>CREATE VIEW</td>
<td>Partial Support</td>
<td>CREATE VIEW is supported as a separate statement from CREATE SCHEMA</td>
<td>CREATE VIEW Statement (Data Definition) [page 904]</td>
</tr>
<tr>
<td>F311-04</td>
<td>CREATE VIEW: WITH CHECK OPTION</td>
<td>Partial Support</td>
<td>CASCADE check is performed on row store. Not supported on column store.</td>
<td>CREATE VIEW Statement (Data Definition) [page 904]</td>
</tr>
<tr>
<td>F311-05</td>
<td>GRANT statement</td>
<td>Partial Support</td>
<td>GRANT is supported as a separate statement from CREATE SCHEMA</td>
<td>GRANT Statement (Access Control) [page 1010]</td>
</tr>
<tr>
<td>F471</td>
<td>Scalar subquery values</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F481</td>
<td>Expanded NULL predicate</td>
<td>Full Support</td>
<td></td>
<td>NULL Predicate [page 74]</td>
</tr>
<tr>
<td>F501</td>
<td>Features and conformance views</td>
<td>Partial Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F501-01</td>
<td>SQL_FEATURES view</td>
<td>Equivalent Support</td>
<td>Supported with a different view.</td>
<td></td>
</tr>
<tr>
<td>F501-02</td>
<td>SQL_SIZING view</td>
<td>No Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F501-03</td>
<td>SQL_LANGUAGES view</td>
<td>No Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F812</td>
<td>Basic flagging</td>
<td>No Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S011</td>
<td>Distinct data types</td>
<td>No Support</td>
<td>User-defined types are not supported.</td>
<td></td>
</tr>
<tr>
<td>T321</td>
<td>Basic SQL-invoked routines</td>
<td>Equivalent Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T321-01</td>
<td>User-defined functions with no overloading</td>
<td>Equivalent Support</td>
<td>The AS clause must be defined after the return type, but before the definition of the routine body.</td>
<td></td>
</tr>
<tr>
<td>T321-02</td>
<td>User-defined stored procedures with no overloading</td>
<td>Equivalent Support</td>
<td>The AS clause must be defined after the return type, but before the definition of the routine body</td>
<td></td>
</tr>
<tr>
<td>T321-03</td>
<td>Function invocation</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T321-04</td>
<td>CALL statement</td>
<td>Full Support</td>
<td></td>
<td>CALL Statement (Procedural) [page 715]</td>
</tr>
<tr>
<td>T321-05</td>
<td>RETURN statement</td>
<td>Full Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T321–06</td>
<td>ROUTINES view</td>
<td>Equivalent Support</td>
<td>Supported with a different view.</td>
<td></td>
</tr>
<tr>
<td>T321–07</td>
<td>PARAMETERS view</td>
<td>Equivalent Support</td>
<td>Supported with a different view.</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Feature</td>
<td>Support</td>
<td>Description</td>
<td>Links</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------------</td>
<td>--------------</td>
<td>-------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>T631</td>
<td>IN predicate with one list element</td>
<td>Full Support</td>
<td>IN Predicate [page 69]</td>
<td></td>
</tr>
</tbody>
</table>
5 SQL Reference for Additional SAP HANA Contexts

This section contains SQL syntax that can be used in additional SAP HANA contexts.

SAP HANA server software and tools are used in many SAP HANA platform and installation scenarios. The feature capability of the SAP HANA server can depend on the type of SAP HANA license as well as any additional capabilities that have been installed separately. Refer to the Feature Scope Description for SAP HANA for your specific SAP HANA version located on the SAP HANA Platform webpage for information about the capabilities available for your license and installation scenario.

⚠️ Caution

This guide contains syntax variations for different product contexts. These are handled as separate reference topics that have titles that announce the context in square brackets at the end of the title. The guide is also organized to keep reference topics together for a context. Always be sure you are using the SQL reference topic specific to your context.

Accelerator for SAP ASE [page 1283]
Dynamic Tiering [page 1289]
Streaming Analytics [page 1452]
Text Mining [page 1455]

5.1 Accelerator for SAP ASE

⚠️ Caution

This guide contains syntax variations for different product contexts. These are handled as separate reference topics that have titles that announce the context in square brackets at the end of the title. The guide is also organized to keep reference topics together for a context. Always be sure you are using the SQL reference topic specific to your context.

ALTER DATABASE Statement [Accelerator for SAP ASE] [page 1284]
Alters an existing database by adding or removing the etsserver service.

ALTER SYSTEM ALTER CONFIGURATION Statement [Accelerator for SAP ASE] [page 1285]
Enables the accelerator for SAP ASE service for provisioning to a tenant database.

ALTER SYSTEM INITIALIZE SERVICE Statement [Accelerator for SAP ASE] [page 1287]
Creates the remote sources and virtual tables, and imports the delivery unit (DU) for the etsservice service.

ALTER SYSTEM UNINITIALIZE SERVICE Statement [Accelerator for SAP ASE] [page 1288]
Removes the remote sources, virtual tables, and the delivery unit (DU) for the etsserver service.
5.1.1 ALTER DATABASE Statement [Accelerator for SAP ASE]

Alters an existing database by adding or removing the etsserver service.

Syntax

```
ALTER DATABASE <database_name>  
  [ ADD 'etsserver' [ AT [LOCATION] '<hostname>[:<port>]' ]     
    | REMOVE 'etsserver' AT [LOCATION] '<hostname>[:<port>]' ]
```

Syntax Elements

- **database_name**
  This parameter specifies the name of the database to which the change is applied.

  ```
  <database_name> ::= <identifier>
  ```

- **hostname**
  Specifies the name of the dedicated accelerator for SAP ASE host.

- **port**
  Specifies the port number where the accelerator for SAP ASE service runs. Port is comprised of 3xx21, where xx is the instance number.

Permissions

You must have the DATABASE ADMIN privilege.

Description

This statement alters an existing database by adding or removing the etsserver service. The use of this statement requires the DATABASE ADMIN privilege.

Examples

The following code example shows you how to add the service etsserver to the database my_database at hostA, for instance 03.

```
ALTER DATABASE my_database ADD 'etsserver' AT LOCATION 'hostA:30321'
```
The following code example shows you how to remove the service etsserver from the database my_database at hostA, for instance 03.

```
ALTER DATABASE my_database REMOVE 'etsserver' AT LOCATION 'hostA:30321'
```

### 5.1.2 ALTER SYSTEM ALTER CONFIGURATION Statement

[Accelerator for SAP ASE]

Enables the accelerator for SAP ASE service for provisioning to a tenant database.

#### Syntax

```
ALTER SYSTEM ALTER CONFIGURATION (<filename>, <layer>[, <layer_name>])     {SET | UNSET} <parameter_key_value_list>','ACCELERATOR_FOR_ASE' [WITH RECONFIGURE]
```

#### Syntax Elements

- **filename**

  ```
  <filename> ::= <string_literal>
  ```

  The filename of the configuration file to be modified. If the file does not exist on the required layer, the file is created when a SET command is used.

- **layer**

  Sets the target layer for the configuration change. This parameter can be either 'SYSTEM', 'HOST' or 'DATABASE'. The SYSTEM layer is the recommended layer for customer settings. The HOST layer should generally only be used for minor configuration, for example parameters contained in daemon.ini. In multiple-container systems, system configuration files have an additional layer DATABASE to facilitate the configuration of properties for individual databases.

  ```
  <layer> ::= <string_literal>
  ```

- **layer_name**

  When `<layer>` is set to 'HOST', `<layer_name>` is used to target either a tenant name or a target host name. For example, 'selxeon12' would target the 'selxeon12' host.

  ```
  <layer_name> ::= <string_literal>
  ```

- **Note**

  The 'HOST' value must be provided in lowercase only.
SET
 Updates the value of a key if the key already exists, or inserts a new key if required.

UNSET
 Removes a key and its associated value.

**parameter_key_value_list**
 A list of configuration file entries to be modified or removed.

```plaintext
<parameter_key_value_list> ::= <parameter_key_value_entry>      
[, <parameter_key_value_entry>]...]
```

**parameter_key_value_entry**
 Specifies the section, key and value of the ini file parameter to be created, modified or removed.

```plaintext
<parameter_key_value_entry> ::= (<section_name>,<parameter_name>)     
[ = <parameter_value>]
```

**section_name**
 The section name of the parameter to be modified.

```plaintext
<section_name> ::= <string_literal>
```

**parameter_name**
 The name of the parameter to be modified.

```plaintext
<parameter_name> ::= <string_literal>
```

**parameter_value**
 The value of the parameter.

```plaintext
<parameter_value> ::= <string_literal>
```

**WITH RECONFIGURE**
 Specifies that the configuration changes are directly applied to the running SAP HANA database instance.

**Permissions**
 You must have the SERVICE ADMIN privilege.

**Description**
 This statement makes the etserver available for provisioning to the tenant databases.
Example

This example sets the parameter ACCELERATOR_FOR_ASE in the customizable_functionalities section of the global.ini file.

```
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'SYSTEM')
SET ('customizable_functionalities','ACCELERATOR_FOR_ASE') = 'true'
```

5.1.3 ALTER SYSTEM INITIALIZE SERVICE Statement

[Accelerator for SAP ASE]

Creates the remote sources and virtual tables, and imports the delivery unit (DU) for the etsservice service.

Syntax

```
ALTER SYSTEM INITIALIZE SERVICE 'etsservice'
    [ AT [ LOCATION ] <hostname>:<port> ]
    WITH CREDENTIAL TYPE 'PASSWORD' USING <accelerator_cred>
```

Syntax Elements

- **hostname** Specifies the name of the dedicated accelerator for SAP ASE host.
- **port** Specifies the port number where the accelerator for SAP ASE service runs. Port is comprised of 3xx21, where xx is the instance number.
- **accelerator_cred**
  Specifies the credential type and the credential information.
  ```
  <accelerator_cred> ::= <string_literal>
  ```

Permissions

You must have the SERVICE ADMIN privilege.
Description

This statement creates the remote sources and virtual tables, and imports the delivery unit (DU) for the service.

Example

This example initialized the etsserver service on host myasehostname, port 30021, with the credentials user = sa and password = MySecretPass. 00 in the port number is the instance number of the system.

```
ALTER SYSTEM INITIALIZE SERVICE 'etsserver' at 'myasehostname:30021'
with CREDENTIAL TYPE 'PASSWORD' USING 'user=sa;password=MySecretPass'
```

5.1.4 ALTER SYSTEM UNINITIALIZE SERVICE Statement
[Accelerator for SAP ASE]

Removes the remote sources, virtual tables, and the delivery unit (DU) for the etsserver service.

Syntax

```
ALTER SYSTEM UNINITIALIZE SERVICE 'etsserver'
  [ AT [ LOCATION] <hostname>:<port> ]
  [ <drop_option> ]
```

Syntax Elements

- **hostname** Specifies the name of the dedicated accelerator for SAP ASE host.
- **port** Specifies the port number where the accelerator for SAP ASE service runs. Port is comprised of 3xx21, where xx is the instance number.
- **drop_option**
  
  Drops the etsserver service.

```
<drop_option> ::= CASCADE | RESTRICT
```

- **CASCADE** Drops the extended service and any dependent objects.
- **RESTRICT**
  
  Drops the extended service only when dependent objects do not exist.
Permissions

You must have the SERVICE ADMIN privilege.

Description

This statement drops the remote sources, virtual tables, and the delivery unit (DU) for the service.

Examples

The following code example drops the remote sources etsserver and dependent objects.

```
ALTER SYSTEM UNINITIALIZE SERVICE 'etsserver' drop cascade;
```

5.2 Dynamic Tiering

⚠️ Caution

SAP HANA dynamic tiering is deprecated. Consider migrating to SAP HANA Native Storage Extension (NSE).

⚠️ Caution

This guide contains syntax variations for different product contexts. These are handled as separate reference topics that have titles that announce the context in square brackets at the end of the title. The guide is also organized to keep reference topics together for a context. Always be sure you are using the SQL reference topic specific to your context.

- ALTER AUDIT POLICY Statement [Dynamic Tiering] [page 1292]
  Enables or disables an audit policy.
- ALTER DATABASE Statement [Dynamic Tiering] [page 1294]
  Alters an existing database by adding or removing a dynamic tiering etsserver service.
- ALTER EXTENDED STORAGE Statement [Dynamic Tiering] [page 1295]
  Lets you manage the extended storage space.
- ALTER INDEX Statement (Multistore Table) [Dynamic Tiering] [page 1298]
  Changes the storage type of an index on a multistore table.
- ALTER STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1299]
  Alters the properties of a data statistics object.
- ALTER STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1305]
Alters the properties of a data statistics object.

**ALTER SYSTEM ALTER CONFIGURATION Statement [Dynamic Tiering] [page 1312]**
Sets or removes configuration parameters in an ini file.

**ALTER SYSTEM RECONFIGURE SERVICE Statement [Dynamic Tiering] [page 1318]**
Reconfigures a specified service by applying the current configuration parameters.

**ALTER TABLE Statement (Extended Store Table) [Dynamic Tiering] [page 1320]**
Changes the definition of an extended store table.

**ALTER TABLE Statement (Multistore Table) [Dynamic Tiering] [page 1326]**
Alters the definition of a multistore table.

**CALL CHECK_CATALOG Statement (Multistore Table) [Dynamic Tiering] [page 1340]**
Performs consistency checks of metadata in both multistore and extended store tables, based on the actions you specify as arguments to the procedure call.

**CALL CHECK_ES Statement (Extended Store Table) [Dynamic Tiering] [page 1343]**
Checks the validity of the current database in the extended store. Optionally, it corrects allocation problems for dbspaces or databases. CHECK_ES does not check a partitioned table if partitioned data exists in offline dbspaces.

**CALL CHECK_TABLE_CONSISTENCY Statement (Multistore Table) [Dynamic Tiering] [page 1346]**
Performs consistency check actions in the column store partition of a multistore table.

**CALL ES_REBUILD_INDEX Statement (Multistore Tables) [Dynamic Tiering] [page 1349]**
When you encounter reports or errors running the CHECK_ES extended store database consistency checker, or if an index fails to meet performance expectations, call this procedure to rebuild indexes for tables in extended storage, including extended store partitions of a multistore table.

**CREATE AUDIT POLICY Statement [Dynamic Tiering] [page 1350]**
Creates a new audit policy for extended tables.

**CREATE EXTENDED STORAGE Statement [Dynamic Tiering] [page 1355]**
Creates a new extended storage database, starts it, and registers it as an SAP HANA service.

**CREATE INDEX Statement (Extended Store Table) [Dynamic Tiering] [page 1357]**
Creates an index on an extended store table column.

**CREATE INDEX Statement (Multistore Table) [Dynamic Tiering] [page 1359]**
Creates an index on a multistore table column.

**CREATE STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1361]**
Creates data statistic objects that allow the query optimizer to make better decisions for query plans.

**CREATE STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1366]**
Creates data statistic objects that allow the query optimizer to make better decisions for query plans.

**CREATE TABLE Statement (Extended Store Table) [Dynamic Tiering] [page 1374]**
Creates a new extended store table in the extended storage.

**CREATE TABLE Statement (Multistore Table) [Dynamic Tiering] [page 1379]**
Creates a new multistore table in the database.

**DROP EXTENDED STORAGE Statement [Dynamic Tiering] [page 1392]**
Removes an existing extended storage configuration from the SAP HANA database.

**DROP STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1393]**
Drops user-defined data statistics objects that the query optimizer uses to make decisions for query plans.

**DROP STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1395]**
Drops user-defined data statistics objects that the query optimizer uses to make decisions for query plans.

**EXPORT Statement (Extended Store Table) [Dynamic Tiering] [page 1399]**
Exports extended store tables, views, column views, synonyms, sequences, and procedures in CSV format.

**EXPORT Statement (Multistore Table) [Dynamic Tiering] [page 1403]**
Exports multistore tables, views, column views, synonyms, sequences, and procedures in CSV format.

**GRANT EXTENDED STORAGE ADMIN Statement [Dynamic Tiering] [page 1407]**
Grants the EXTENDED STORAGE ADMIN system privilege required to manage dynamic tiering and create extended storage.

**IMPORT Statement (Extended Store Table) [Dynamic Tiering] [page 1409]**
Imports catalog objects into an extended store table.

**IMPORT Statement (Multistore Table) [Dynamic Tiering] [page 1414]**
Imports catalog objects into a multistore table.

**IMPORT FROM Statement (Extended Store Table) [Dynamic Tiering] [page 1418]**
Imports external data from a file into an existing extended store table.

**IMPORT FROM Statement (Multistore Table) [Dynamic Tiering] [page 1424]**
Imports external data from a file into existing extended storage partitions in a multistore table.

**INSERT Statement (Extended Store Table) [Dynamic Tiering] [page 1430]**
Adds a record to an extended store table.

**INSERT Statement (Multistore Table) [Dynamic Tiering] [page 1433]**

**MERGE DELTA Statement [Dynamic Tiering] [page 1435]**
On a delta-enabled extended store or multistore table, MERGE DELTA triggers a merge of data from the delta dbspace to the user dbspace. When used in a multistore table, you can restrict a manual delta merge to just the default store partition or extended store partition of the multistore table.

**REFRESH STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1437]**
Refreshes data statistic objects that the query optimizer uses to make better decisions for query plans.

**REFRESH STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1441]**
Refreshes data statistic objects that the query optimizer uses to make better decisions for query plans.

**UPDATE Statement (Extended Store Table) [Dynamic Tiering] [page 1445]**
Changes the values of the records of an extended store table.

**UPDATE Statement (Multistore Table) [Dynamic Tiering] [page 1448]**
Changes the values of the records of a multistore table.

**Additional Syntax [Dynamic Tiering] [page 1451]**
SQL syntax that matches SAP HANA core syntax.
5.2.1 ALTER AUDIT POLICY Statement [Dynamic Tiering]

Enables or disables an audit policy.

**Syntax**

```
ALTER AUDIT POLICY <policy_name> [     <audit_mode>      | <set_audit_trail_type>      | <reset_audit_trail_type>
```

**Syntax Elements**

- **policy_name**
  Specifies the name of the audit policy to be altered.
  ```
  <policy_name> ::= <identifier>
  ```

- **audit_mode**
  Enables or disables the audit policy.
  ```
  <audit_mode> ::= ENABLE | DISABLE
  
  ENABLE
  Enables the audit policy.
  DISABLE
  Disables the audit policy.
  ```

- **set_audit_trail_type**
  Specifies the audit trail target(s) for the audit policy.
  ```
  <set_audit_trail_type> ::= SET TRAIL TYPE <audit_trail_type_list>
  <audit_trail_type_list> ::= <audit_trail_type_name>[,<audit_trail_type_name>[,...]]
  <audit_trail_type_name> ::= SYSLOG
  ```

- **reset_audit_trail_type**
  Resets the audit trail target type(s) to the system default.
  ```
  <reset_audit_trail_type> ::= RESET TRAIL TYPE
  ```
Permissions

You must have the AUDIT ADMIN privilege.

Description

The ALTER AUDIT POLICY statement enables or disables an audit policy. `<policy_name>` must specify an existing audit policy. Only database users with the system privilege AUDIT ADMIN are allowed to alter an audit policy. Users with this privilege can alter any audit policy, regardless of if they are the creator of the policy.

When an audit policy is created, it is created in the disabled state. Therefore the audit policy has to be enabled to make its audit actions take effect.

An audit policy can be disabled and enabled as often as required.

One or more audit trail targets could be specified for an audit policy at the time of creation (see CREATE AUDIT POLICY Statement [Dynamic Tiering] [page 1350]) or after creation.

The allowed audit trail targets are:

- SYSLOG: uses the system syslog
- TABLE: stores audit information in database table. The audit log is accessible using AUDIT_LOG system view
- CSV: stores audit information as comma-separated values in a text file. Should be used only for testing purposes.

Example

For this example, you need to first create an audit policy called `ext_store_admin_policy` using the following statement.

```
CREATE AUDIT POLICY ext_store_admin_policy CREATE EXTENDED STORAGE, ALTER
EXTENDED STORAGE, DROP EXTENDED STORAGE LEVEL CRITICAL
```

You enable the `ext_store_admin_policy` policy.

```
ALTER AUDIT POLICY ext_store_admin_policy ENABLE;
```

You disable the `ext_store_admin_policy` audit policy.

```
ALTER AUDIT POLICY ext_store_admin_policy DISABLE;
```

Related Information

AUDIT_POLICIES System View [page 1503]
5.2.2 ALTER DATABASE Statement [Dynamic Tiering]

Alters an existing database by adding or removing a dynamic tiering esserver service.

Syntax

```
ALTER DATABASE <database_name>
    ADD 'esserver' [AT [LOCATION] [ '<hostname>[:<port>]'] ]
| REMOVE 'esserver' AT [LOCATION] '<hostname>:<port>'
```

Syntax Elements

- **database_name**: Specifies the name of the database to which the service is applied.

  ```
  <database_name> ::= <identifier>
  ```

- **hostname**: Specifies the name of the dedicated dynamic tiering host.

- **port**: Specifies the port number where the dynamic tiering service runs. Port is comprised of 3xx16, where xx is the instance number.

Permissions

You must have the DATABASE ADMIN privilege.

Description

This statement can only be executed on the SYSTEM database (systemDB).
Examples

The following code example shows you how to add the service esserver to the database my_database at hostA, for instance 03.

```
ALTER DATABASE my_database ADD 'esserver' AT LOCATION 'hostA:30312'
```

The following code example shows you how to remove the service esserver from the database my_database at hostA, for instance 03.

```
ALTER DATABASE my_database REMOVE 'esserver' AT LOCATION 'hostA:30312'
```

5.2.3 ALTER EXTENDED STORAGE Statement [Dynamic Tiering]

Lets you manage the extended storage space.

Syntax

```
ALTER EXTENDED STORAGE [<enable_delta_clause>] [<add_dbfile_clause>]       [<empty_dbfile_clause>] [<drop_dbfile_clause>]       [<increase_dbfile_size_clause>]
```

Syntax Elements

**enable_delta_clause**

Create a delta dbspace or disable delta dbspace creation in extended storage. Delta dbspaces are optimized for write access. ENABLE DELTA creates a dbspace named ES_DELTA of size MIN (DELTA_MEMORY_MB, size of ES_USER dbspace) to use row-level versioning in your extended storage. You may add files to ES_DELTA later as needed.

```
<enable_delta_clause> ::=  {ENABLE | DISABLE}  DELTA
```

**add_dbfile_clause**

Add one or more dbspace files into specified dbspace.

```
<add_dbfile_clause> ::= ALTER DBSPACE <dbspace_name> ADD <new_file_spec> [,                <new_file_spec> ...]
```

dbspace_name
Name of the dbspace.

<dbname> ::= <identifier>

new_file_spec

Name of the new dbfile.

<new_file_spec> ::= FILE <logical_filename> 'file-path' <file_opts>

logical_filename

Logical undelimited name of the dbfile.

<logical_filename> ::= <identifier>

file_opts

Options of the dbfile.

<file_opts> ::= [[SIZE] <file_size> {KB|MB|GB|TB} [RESERVE <reserve_size> {KB|MB|GB|TB}]]

file_size

Size of the dbfile. The size of any individual dbfile added to a dynamic tiering database cannot exceed 1 TB.

<file_size> ::= <unsigned_integer>

reserve_size

Reserved size of the dbfile. You can use the reserved space of a dbfile to increase its size.

<reserve_size> ::= <unsigned_integer>

Default values are:

- MB is the default unit for file_size and reserve_size.
- The default value for file_size is 8 MB.
- The default value for reserve_size is 0 MB.

empty_dbfile_clause

Empty one or more files from specified dbspace. You must empty files before you can drop them. You can only empty one file per statement.

<empty_dbfile_clause> ::= ALTER DBSPACE <dbname> EMPTY <empty_file_spec>

empty_file_spec

Name of the dbfile being emptied.

<empty_file_spec> ::= FILE <logical_filename>

drop_dbfile_clause
Drop one or more dbfiles from specified dbspace.

\[
\text{<drop_dbfile_clause>} ::= \text{ALTER DBSPACE } \text{<dbspace_name>} \text{ DROP } \text{<drop_file_spec>} \\
[\text{[, <drop_file_spec>} ...]
\]

\text{drop_file_spec}

Name of the dbfile being dropped.

\[
\text{<drop_file_spec>} ::= \text{FILE } \text{<logical_filename>}
\]

\text{increase_dbfile_size_clause}

Increase the size of a dbfile for the specified dbspace. The \text{ALTER FILE ADD} clause uses dbfile reserved space to enlarge the file. Make sure that there is adequate reserved space before issuing this statement.

\[
\text{<increase_dbfile_size_clause>} ::= \text{ALTER DBSPACE } \text{<dbspace_name>} \text{ ALTER FILE } \\
\text{<logical_filename>} \text{ ADD } \text{<add_file_size>} \{\text{KB} | \text{MB} | \text{GB} | \text{TB}\}
\]

\text{add_file_size}

New size of the dbfile.

\[
\text{<add_file_size>} ::= \text{<unsigned_integer>}
\]

\text{Permissions}

You must have the \text{EXTENDED STORAGE ADMIN} privilege.

\text{Description}

The \text{ALTER EXTENDED STORAGE} statement lets you manage the extended storage space.

\text{Example}

You add a delta dbspace called \text{ES_DELTA} to extended storage.

\[
\text{ALTER EXTENDED STORAGE ENABLE DELTA.}
\]

You add one or more new dbfiles into dbspace \text{ES_USER}.

\[
\begin{align*}
\text{ALTER EXTENDED STORAGE ALTER DBSPACE } \text{ES_USER} \text{ ADD FILE } f_1 \ 'f_1\_file' \\
\text{SIZE 100 MB}; \\
\text{ALTER EXTENDED STORAGE ALTER DBSPACE } \text{ES_USER} \text{ ADD FILE } f_2 \ 'f_2\_file.db' \text{ SIZE 200 MB,} \\
\text{FILE } f_3 \ 'f_3\_file.dat' \text{ SIZE 100 MB RESERVE 50 MB;}
\end{align*}
\]
You empty and drop one or more dbfiles from dbspace ES_USER.

```
ALTER EXTENDED STORAGE ALTER DBSPACE ES_USER EMPTY FILE f1;
ALTER EXTENDED STORAGE ALTER DBSPACE ES_USER EMPTY FILE f2;
ALTER EXTENDED STORAGE ALTER DBSPACE ES_USER EMPTY FILE f3;
ALTER EXTENDED STORAGE ALTER DBSPACE ES_USER EMPTY FILE f4;
ALTER EXTENDED STORAGE ALTER DBSPACE ES_USER DROP FILE f1, FILE f2, FILE f3, FILE f4;
```

You increase the size of dbfile f4. This statement uses 10MB from the dbfile reserved space to increase the dbfile size.

```
ALTER EXTENDED STORAGE ALTER DBSPACE ES_USER ALTER FILE f4 ADD 10 MB;
```

### 5.2.4 ALTER INDEX Statement (Multistore Table) [Dynamic Tiering]

Changes the storage type of an index on a multistore table.

#### Syntax

```
ALTER INDEX <index_name>   {CREATE|DROP} FOR {DEFAULT|EXTENDED} STORAGE
```

#### Syntax Elements

- **index_name**
  
  Specifies the name of the index with an optional schema name.

  ```
  <index_name> ::= <schema_name>.<identifier>  
  <schema_name> ::= <unicode_name>
  ```

#### Description

The ALTER INDEX statement changes the storage type of the index on a multistore table.
The CREATE FOR clause adds the index to default or extended storage. The DROP FOR clause drops the index from the specified (extended or default) storage.

Permissions

This statement requires the INDEX object privilege.

Example

Create multistore table ext_a:

```sql
CREATE COLUMN TABLE EXT_A (c1 INT, c2 INT)  PARTITION BY RANGE (c1) (USING DEFAULT STORAGE (PARTITION 0 <= values < 10000, PARTITION 10000 <= values < 1000000) USING EXTENDED STORAGE (PARTITION 1000000 <= values < 100000000));
```

Create a partial index on the column store only:

```sql
CREATE BTREE INDEX idx_a ON ext_a (c1) FOR DEFAULT STORAGE;
```

Alter the index storage type to create it also on extended storage:

```sql
ALTER INDEX idx_a CREATE FOR EXTENDED STORAGE;
```

Change the index storage type to remove it from the column store. Now it is a partial index on extended storage only:

```sql
ALTER INDEX idx_a DROP FOR DEFAULT STORAGE;
```

5.2.5 ALTER STATISTICS Statement (Extended Store Table) [Dynamic Tiering]

Alters the properties of a data statistics object.

Syntax

```sql
ALTER STATISTICS { <data_statistics_name>[, <data_statistics_name>[,...] ] }  
| ON <data_sources> [ [ HAVING ] <match_properties> ]  
[ SET <set_data_statistics_properties> ] 
[ <add_drop_data_statistics_properties> ]  
[ <initial_refresh> ];
```
Syntax Element

data_statistics_name
Specifies the name of the data statistics object.

```
<data_statistics_name> ::= [<schema_name>].<identifier>
<schema_name> ::= <identifier>
```

data_sources
Specifies the data source(s) of the data statistics objects.

```
<data_sources> ::= <table_name> [ ( <column_name>[, <column_name> [... ] ) ] [ <match_type> ]
```

For RECORD COUNT data statistics objects, you cannot specify columns as part of data_sources.

table_name
Specifies the table on which the data statistics are defined.

```
<table_name> ::= [<schema_name>].<identifier>
```

column_name
Specifies the column for which the data statistics are defined.

```
<column_name> ::= <identifier>
```

If no <column_name> is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

match_type
Controls which data statistics objects to match to <data_sources>.

```
<match_type> ::= EXACT | CASCADE
```

If <match_type> is not specified, then any data statistics object(s) that reference all or some of the columns, but no other columns specified in <data_sources> are refreshed.

Specify EXACT to refresh data statistics objects that precisely match <data_sources> (including column order).

Specify CASCADE to refresh data statistics objects that reference at least one column in <data_sources>

Use this table to understand how matching is performed based on <match_type> when <data_sources> is T(A, B, C):

<table>
<thead>
<tr>
<th>Match type</th>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>(not specified)</td>
<td>T(A,C)</td>
<td>T(A,X) - because T(X) is not a column in &lt;data_sources&gt;.</td>
</tr>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td>Match type</td>
<td>Example matches</td>
<td>Example non-matches</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>EXACT</td>
<td>T(A,B,C)</td>
<td>T(B,A,C) - because the column order is different than the column order of &lt;data_sources&gt;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T(A) - because it does not contain the exact same columns and column order of &lt;data_sources&gt;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T(A,B,C) - because T(X) is not a column in &lt;data_sources&gt;.</td>
</tr>
<tr>
<td>CASCADE</td>
<td>T(A,C)</td>
<td>T(X) - because it does not contain any columns that match the columns in &lt;data_sources&gt;.</td>
</tr>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A,B,C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A,X)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(C,B,A,X)</td>
<td></td>
</tr>
</tbody>
</table>

**match_properties**

Specifies properties to use for matching when selecting data statistics.

```
<match_properties> ::= <match_property>[...]
<match_property> ::= TYPE <data_statistics_type> | REFRESH TYPE <refresh_type_filter>
```

If TYPE is not specified, then all data statistics objects of any type on the specified data sources are refreshed (ALL). For descriptions of the supported data statistics types see the CREATE STATISTICS Statement (Extended Store Table) topic.

**data_statistics_type**

Specifies the type of data statistics objects to match when selecting the data statistics.

```
<data_statistics_type> ::= TYPE <type_name>
[type_name] ::= HISTOGRAM | SIMPLE | TOPK | SKETCH | SAMPLE | RECORD COUNT | ALL
```

**refresh_type_filter**
Specifies the refresh strategy to match in the data statistics objects when selecting data statistics. ALL is the default.

```
<refresh_type_filter> ::= AUTO | MANUAL | ALL
```

**set_data_statistics_properties**

Specifies the properties of the data statistics objects to modify.

```
<set_data_statistics_properties> :=
<data_statistics_property>[, <data_statistics_property>][,...]
<data_statistics_property> ::= REFRESH TYPE <refresh_type>
   ENABLE <on_off>
   BUCKETS <unsigned_integer>
   { MEMORY <memory_bytes> | MEMORY PERCENT <memory_percentage> }
```

**REFRESH TYPE refresh_type**

Specifies the strategy for the data statistics object.

```
<refresh_type> ::= { AUTO | MANUAL | DEFAULT }
```

AUTO specifies that the data statistics object is refreshed automatically when underlying data changes. AUTO is only supported on column store, extended store, and multistore tables.

MANUAL specifies that the database statistics object is not refreshed until a rebuild is explicitly requested by a REFRESH STATISTICS statement.

DEFAULT specifies that the database server decides the best refresh strategy based on the data source. For example, for data statistics objects on column store data sources, the database server applies AUTO for the default.

REFRESH TYPE only affects data statistics objects that are enabled.

**ENABLE on_off**

Controls whether the optimizer uses the data statistics object.

```
<on_off> ::= { ON | OFF }
```

ENABLE ON enables the optimizer to see the data statistics object. The data statistics object must be populated with data for the optimizer to use it.

ENABLE OFF disables the use of the data statistics object by the optimizer and prevents the ability to refresh the data statistics object. Data statistics objects that are not enabled can still be dropped. To make a data statistics object with ENABLE OFF accessible to the optimizer, execute an ALTER STATISTICS...ENABLE ON statement.

**BUCKETS unsigned_integer**

The BUCKETS property is only for use with TYPE HISTOGRAM or TOPK. For HISTOGRAM, BUCKETS specifies the maximum number of data buckets in the HISTOGRAM. For TOPK, BUCKETS specifies the K value.

The default is automatically determined by the data statistics building algorithm in use.

If a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.
For column store, extended store, and multistore tables only, if a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

**MEMORY memory_bytes**

Specifies the maximum amount of memory, in bytes, to use for QOPTIMAL HISTOGRAMS.

\[
<\text{memory_bytes}> ::= <\text{unsigned_integer}>
\]

The MEMORY parameter limits the memory for QOPTIMAL HISTOGRAMS. MEMORY applies only to the QOPTIMAL HISTOGRAM algorithm. Small values for MEMORY may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

**MEMORY PERCENT memory_percentage**

Specifies the maximum amount of memory to use for the data statistics object, expressed as a percentage of the space used by the data source.

\[
<\text{memory_percentage}> ::= <\text{unsigned_integer}>
\]

HISTOGRAMS can use a large amount of memory for some data sources. \(<\text{memory_percentage}>\) represents the maximum amount of memory that can be used for the data statistics object. For example, if a data source is a table column that uses 100 MB of memory, and \(<\text{memory_percentage}>\) is 5, then the data statistics object for this column can use, at most, 5 MB for its in-memory representation.

The default is automatically determined by the HISTOGRAM algorithm in use. Small values for MEMORY PERCENT may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

**initial_refresh**

Specifies whether to repopulate the data statistics object with data after altering it.

\[
<\text{initial_refresh}> ::= [ \text{NO} ] \text{INITIAL REFRESH}
\]

If the object was built, then disabled, and is now being re-enabled, then initial refresh is not required.

**INITIAL REFRESH**

Alters the definition of the data statistics object and repopulates it with data. The default behavior is INITIAL REFRESH.

**NO INITIAL REFRESH**

Alters the definition of the data statistics object, but does not repopulate it with data.

Use NO INITIAL REFRESH when you want to change the underlying data before refreshing the data statistics object.

You cannot specify NO INITIAL REFRESH if ENABLE OFF is not specified.

**Permissions**

One of the following is true:
You own the object you are altering statistics on.

You have the ALTER privilege on the object, which was created by another user and you are altering statistics on.

**Description**

The ALTER STATISTICS statement alters the properties of a data statistics object. A typical change to a data statistics object might be to enable or disable it, or to change settings for constraints such as BUCKETS.

You may specify the ADD, DROP, and SET clauses in any order in the ALTER STATISTICS statement, but not more than once each.

Refer to the CREATE STATISTICS statement for complete descriptions of the data statistic types `<data_statistics_type>`.

You cannot alter the type for the data statistics object. For example, you cannot change a data statistics object from a HISTOGRAM to a SIMPLE; you must create the SIMPLE data statistics object separately.

**Examples**

Set the refresh type of HISTOGRAM on data source T(X) to AUTO and rebuilds the data statistics object:

```
ALTER STATISTICS on MYSHEMA.T(X) TYPE HISTOGRAM SET REFRESH TYPE AUTO;
```

Set the refresh type and number of buckets for two HISTOGRAMS on table T, specifying HISTOGRAMS by name and rebuilds the data statistics object:

```
ALTER STATISTICS HIST1_T, HIST2_T SET BUCKETS 100 REFRESH TYPE AUTO;
```

**Related Information**

- CREATE STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1361]
- REFRESH STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1437]
- DROP STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1393]
- M_DATA_STATISTICS System View [page 1851]
- M_SYSTEM_DATA_STATISTICS System View [page 2189]
5.2.6 ALTER STATISTICS Statement (Multistore Table)
[Dynamic Tiering]

Alters the properties of a data statistics object.

Syntax

```
ALTER STATISTICS { <data_statistics_name>[,...] | ON <data_sources> [ [ HAVING ]
<match_properties> ] } [ SET <set_data_statistics_properties> ] [ <add_drop_data_statistics_properties> ] [ <initial_refresh> ];
```

Syntax Element

data_statistics_name

Specifies the name of the data statistics object.

```
<data_statistics_name> ::= [<schema_name>.]<identifier>
<schema_name> ::= <identifier>
```

data_sources

Specifies the data source(s) of the data statistics objects.

```
<data_sources> ::= <table_name> [ ( <column_name>[, <column_name>[,...] ] )]
[ FOR { DEFAULT | EXTENDED } STORAGE ] [ <match_type> ]
```

For RECORD COUNT data statistics objects, you cannot specify columns as part of data_sources.

table_name

Specifies the table on which the data statistics are defined.

```
<table_name> ::= [<schema_name>.]<identifier>
```

column_name

Specifies the column for which the data statistics are defined.

```
<column_name> ::= <identifier>
```

If no `<column_name>` is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

FOR {DEFAULT | EXTENDED} STORAGE

The STORAGE clause can be specified only for multistore tables. Specify DEFAULT when the data statistics object is created on the column store part. Specify EXTENDED STORAGE when the data
statistics object is created on the extended storage part. If you do not specify anything, then two data
statistics objects are created, one on the extended storage part and one on the column storage part.

Statistics of some types can only be created on one storage class partition; for example, SKETCH on
HANA column store partitions, and RECORD COUNT on extended storage partitions. Attempting to
create such objects on a multistore table without specifying storage type returns an error.

You can implement different statistics types for different storage classes, and create statistics to
address specific performance problems that only affect the query plan for one of the storage classes of
the table. The statement returns an error if no partitions of the specified storage type exist.

c
match_type

Controls which data statistics objects to match to <data_sources>.

<match_type> ::= EXACT | CASCADE

If <match_type> is not specified, then any data statistics object(s) that reference all or some of the
columns, but no other columns specified in <data_sources> are refreshed.

Specify EXACT to refresh data statistics objects that precisely match <data_sources> (including
column order).

Specify CASCADE to refresh data statistics objects that reference at least one column in
<data_sources>.

Use this table to understand how matching is performed based on <match_type> when
<data_sources> is T(A, B, C):

<table>
<thead>
<tr>
<th>Match type</th>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>(not specified)</td>
<td>T(A,C)</td>
<td>T(A,X) - because T(X) is not a column in &lt;data_sources&gt;.</td>
</tr>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,A)</td>
<td></td>
</tr>
<tr>
<td>EXACT</td>
<td>T(A,B,C)</td>
<td>T(B,A,C) - because the column order is different than the column order of &lt;data_sources&gt;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T(A) - because it does not contain the exact same columns and column order of &lt;data_sources&gt;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T(X,A,B,C) - because T(X) is not a column in &lt;data_sources&gt;.</td>
</tr>
<tr>
<td>CASCADE</td>
<td>T(A,C)</td>
<td>T(X) - because it does not contain any columns that match the columns in &lt;data_sources&gt;.</td>
</tr>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A,B,C)</td>
<td></td>
</tr>
</tbody>
</table>
### match_properties

Specifies properties to use for matching when selecting data statistics.

```sql
<match_properties> ::= <match_property>[...]
<match_properties> ::= TYPE <data_statistics_type>  | REFRESH TYPE <refresh_type_filter>
```

If TYPE is not specified, then all data statistics objects of any type on the specified data sources are altered (ALL). For descriptions of the supported data statistics types see the `CREATE STATISTICS Statement (Multistore Table)` topic.

### data_statistics_type

Specifies the type of data statistics objects to match when selecting the data statistics.

```sql
<data_statistics_type> ::= TYPE <type_name>
<type_name> ::= HISTOGRAM | SIMPLE | TOPK | SKETCH | SAMPLE | RECORD COUNT | ALL
```

### refresh_type_filter

Specifies the refresh strategy to match in the data statistics objects when selecting data statistics. ALL is the default.

```sql
<refresh_type_filter> ::= AUTO | MANUAL | ALL
```

### set_data_statistics_properties

Specifies the properties of the data statistics objects to modify.

```sql
<set_data_statistics_properties> ::=<data_statistics_property>[, <data_statistics_property>[,...] ]
<data_statistics_properties> ::= REFRESH TYPE <refresh_type>
| ENABLE <on_off>
| QERROR <numeric_literal>
| QTHETA <unsigned_integer>
| { MEMORY <memory_bytes>  | MEMORY PERCENT <memory_percentage> } 
| PERSISTENT <on_off>
| CONSTRAINT <constraint_param>
```

### REFRESH TYPE refresh_type
Specifies the strategy for the data statistics object.

```plaintext
<refresh_type> ::= { AUTO | MANUAL | DEFAULT }
```

AUTO specifies that the data statistics object is refreshed automatically when underlying data changes. AUTO is only supported on column store, extended store, and multistore tables.

MANUAL specifies that the database statistics object is not refreshed until a rebuild is explicitly requested by a REFRESH STATISTICS statement.

DEFAULT specifies that the database server decides the best refresh strategy based on the data source. For example, for data statistics objects on column store data sources, the database server applies AUTO for the default.

REFRESH TYPE only affects data statistics objects that are enabled.

**ENABLE on_off**

Controls whether the optimizer uses the data statistics object.

```plaintext
<on_off> ::= { ON | OFF }
```

ENABLE ON enables the optimizer to see the data statistics object. The data statistics object must be populated with data for the optimizer to use it.

ENABLE OFF disables the use of the data statistics object by the optimizer and prevents the ability to refresh the data statistics object. Data statistics objects that are not enabled can still be dropped. To make a data statistics object with ENABLE OFF accessible to the optimizer, execute an ALTER STATISTICS...ENABLE ON statement.

**BUCKETS unsigned_integer**

The BUCKETS property is only for use with TYPE HISTOGRAM or TOPK. For HISTOGRAM, BUCKETS specifies the maximum number of data buckets in the HISTOGRAM. For TOPK, BUCKETS specifies the K value.

The default is automatically determined by the data statistics building algorithm in use.

If a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

For column store, extended store, and multistore tables only, if a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

**QERROR numeric_literal**

Specifies the Q error to use for the q-optimal HISTOGRAM. You can specify this parameter when TYPE is HISTOGRAM and CONSTRAINT is QOPTIMAL. The default is automatically determined by the HISTOGRAM algorithm in use.

**QTHETA unsigned_integer**

Specifies a lower bound on the frequencies for which a q error constraint is applied for a q-optimal HISTOGRAM. You can specify this parameter when TYPE is HISTOGRAM and CONSTRAINT is QOPTIMAL. The default is automatically determined by the HISTOGRAM algorithm in use.

**MEMORY memory_bytes**
Specifies the maximum amount of memory, in bytes, to use for QOPTIMAL HISTOGRAMS.

\[
\text{<memory_bytes>} ::= \text{<unsigned_integer>}
\]

The MEMORY parameter limits the memory for QOPTIMAL HISTOGRAMS. MEMORY applies only to the QOPTIMAL HISTOGRAM algorithm. Small values for MEMORY may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

**MEMORY PERCENT memory_percentage**

Specifies the maximum amount of memory to use for the data statistics object, expressed as a percentage of the space used by the data source.

\[
\text{<memory_percentage>} ::= \text{<unsigned_integer>}
\]

HISTOGRAMS can use a large amount of memory for some data sources. \(<\text{memory_percentage}>\) represents the maximum amount of memory that can be used for the data statistics object. For example, if a data source is a table column that uses 100 MB of memory, and \(<\text{memory_percentage}>\) is 5, then the data statistics object for this column can use, at most, 5 MB for its in-memory representation.

The default is automatically determined by the HISTOGRAM algorithm in use. Small values for MEMORY PERCENT may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

**PERSISTENT on_off**

Specifies whether data statistics object data persists in the storage of the table, and only applies to QOPTIMAL HISTOGRAMS on column store tables. The default is PERSISTENT ON.

\[
\text{<on_off>} ::= \{ \text{ON} | \text{OFF} \}
\]

Other statistics types are always persistent.

**CONSTRAINT constraint_param**

Specifies the mathematical constraint used to build HISTOGRAM or SKETCH.

\[
\text{CONSTRAINT} \ <\text{constraint_param}>
\]

- HISTOGRAM \(<\text{constraint_param}>\) specifies the mathematical constraint for the HISTOGRAM:

\[
\text{<constraint_param>} ::= \text{QOPTIMAL} | \text{MAXDIFF}
\]

- QOPTIMAL – For column store tables, the default value of \(<\text{constraint_param}>\) for the column storage part of the table is QOPTIMAL.
- MAXDIFF – For extended tables, and extended partitions of multistore tables, the default value of \(<\text{constraint_param}>\) for the extended storage part of the table is MAXDIFF.

**i Note**

HISTOGRAM sizing restrictions (BUCKETS, MEMORY, and MEMORY PERCENT) are applied per HISTOGRAM. A non-default CONSTRAINT for a HISTOGRAM returns an error.
SKETCH constraints apply only to the column store data statistics, not to the extended storage part of the table. `<constraint_param>` specifies the algorithm to use to build the SKETCH. The default is LOGLOGCOUNTING; the remaining algorithms are for internal use.

```
<constraint_param> ::=  
KMINVAL  
PCSA  
LINEARCOUNTING  
LOGCOUNTING  
LOGLOGCOUNTING  
SUPERLOGLOGCOUNTING
```

**add_drop_data_statistics_properties**

Specifies the properties of the data statistics objects you set using the ADD and DROP keywords.

```
<add_drop_data_statistics_properties> ::= <add_drop_property> [  
<add_drop_property> [...]  
]  
<add_drop_property> ::=  
{ ADD | DROP } VALID FOR <valid_for_list>
```

This property is supported for column store, extended store, and multistore tables, but not virtual tables or linked database.

**VALID FOR valid_for_list**

Defines how the data statistics object may be used. The VALID FOR clause is only permitted with statistics type SIMPLE.

```
<valid_for_list> ::= <usage>[, <usage>[,...] ]  
<usage> ::= { ESTIMATION | DATA DEPENDENCY }
```

**ESTIMATION** initializes the data statistics object for use by the optimizer to improve selectivity estimation. SIMPLE data statistics objects are initialized for estimation use by default.

**DATA DEPENDENCY** applies to partitioned column store and multistore tables only. It initializes the data statistics object to be used by features that require higher (or more) data consistency (including automatically refreshing and rebuilding when needed), such as the dynamic partition pruning feature. For more information about the dynamic partition pruning feature, including the types of columns that can have data statistics defined for dynamic partition pruning, see the *SAP HANA Administration Guide*.

**initial_refresh**

Specifies whether to repopulate the data statistics object with data after altering it.

```
<initial_refresh> ::= [ NO ] INITIAL REFRESH
```

If the object was built, then disabled, and is now being re-enabled, then initial refresh is not required.

**INITIAL REFRESH**

Alters the definition of the data statistics object and repopulates it with data. The default behavior is INITIAL REFRESH.

**NO INITIAL REFRESH**

Alters the definition of the data statistics object, but does not repopulate it with data.

Use NO INITIAL REFRESH when you want to change the underlying data before refreshing the data statistics object.
You cannot specify NO INITIAL REFRESH if ENABLE OFF is not specified.

Permissions

One of the following is true:

• You own the object you are altering statistics on.
• You have the ALTER privilege on the object, which was created by another user and you are altering statistics on.

Description

The ALTER STATISTICS statement alters the properties of a data statistics object. A typical change to a data statistics object might be to enable or disable it, or to change settings for constraints such as BUCKETS or ENABLE ON/OFF.

Refer to the CREATE STATISTICS statement for complete descriptions of the data statistic types (<data_statistics_type>).

You cannot alter the type for the data statistics object. For example, you cannot change a data statistics object from a HISTOGRAM to a SIMPLE, you must create the SIMPLE data statistics object separately.

Example

Set the refresh type of HISTOGRAMS on data source T(X) to AUTO and rebuilds the data statistics object:

```
ALTER STATISTICS on MYSYSTEM.T(X) TYPE HISTOGRAM SET REFRESH TYPE AUTO;
```

Set the refresh type and number of buckets for two HISTOGRAMS on table T, specifying histograms by name and rebuilds the data statistics object:

```
ALTER STATISTICS HIST1_T, HIST2_T SET BUCKETS 100 REFRESH TYPE AUTO;
```

Related Information

- REFRESH STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1441]
- CREATE STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1366]
- DROP STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1395]
- M_DATA_STATISTICS System View [page 1851]
- M_SYSTEM_DATA_STATISTICS System View [page 2189]
5.2.7 ALTER SYSTEM ALTER CONFIGURATION Statement
[Dynamic Tiering]

Sets or removes configuration parameters in an ini file.

Syntax

```
ALTER SYSTEM ALTER CONFIGURATION (<filename>, <layer>[, <layer_name>])     {SET | UNSET} <parameter_key_value_list> [WITH RECONFIGURE]
```

Syntax Elements

- **filename**
  The filename of the configuration file to be modified. If the file does not exist on the required layer, the file is created when a SET command is used.

  `<filename> ::= <string_literal>`

- **layer**
  Sets the target layer for the configuration change. This parameter can be ‘SYSTEM’, ‘HOST’ or ‘DATABASE’. The SYSTEM layer is the recommended layer for customer settings. The HOST layer should generally only be used for minor configuration, for example parameters contained in daemon.ini. In multitenant systems, system configuration files have an additional layer DATABASE to facilitate the configuration of properties for individual databases.

  `<layer> ::= <string_literal>`

- **layer_name**
  If the layer parameter above is set to ‘HOST’, `<layer_name>` is used to target either a tenant name or a target host name. For example, ‘selexon12’ would target the ‘selexon12’ host.

  `<layer_name> ::= <string_literal>`

- **Note**
  The ‘HOST’ value must be provided in lowercase only.

- **SET**
  Updates the value of a key if the key already exists, or inserts a new key if required.

- **UNSET**
  Removes a key and its associated value.

- **parameter_key_value_list**
  The set of key-value pairs to be set or unset.
A list of configuration file entries to be modified or removed.

```<parameter_key_value_list> ::= <parameter_key_value_entry> [{, <parameter_key_value_entry>}...]
```

**parameter_key_value_entry**

Specifies the section, key and value of the ini file parameter to be created, modified or removed.

```<parameter_key_value_entry> ::= (<section_name>,<parameter_name>) [ = <parameter_value>]
```

**section_name**

Specifies the section name of the parameter to be modified.

```<section_name> ::= <string_literal>
```

**parameter_name**

Specifies the name of the parameter to be modified.

```<parameter_name> ::= <string_literal>
```

**parameter_value**

Specifies the value of the parameter.

```<parameter_value> ::= <string_literal>
```

**WITH RECONFIGURE**

Specifies that the configuration changes are directly applied to the running SAP HANA database instance.

When WITH RECONFIGURE is not specified the configuration changes are written to the required ini file, however the modified values are not applied to the current running system. The changes are only applied during a restart of the SAP HANA database or a subsequent configuration change with WITH RECONFIGURE. In this case there can be inconsistencies between the ini file contents and the actual configuration value that the SAP HANA database is currently using.

**Permissions**

You must have the EXTENDED STORAGE ADMIN and INIFILE ADMIN system privileges.

**Description**

Sets or removes configuration parameters in an ini file. Ini file configuration is used for the layered configuration of DEFAULT, SYSTEM, HOST layers.

**Note**

The DEFAULT layer configuration parameters cannot be changed or removed using this command.
The following is an example of ini file locations:

- **DEFAULT**: /usr/sap/<SYSTEMNAME>/HDB<INSTANCENUMBER>/exe/config/indexserver.ini
- **SYSTEM**: /usr/sap/<SYSTEMNAME>/SYS/global/hdb/custom/config/indexserver.ini
- **HOST**: /usr/sap/<SYSTEMNAME>/HDB<INSTANCENUMBER>/<HOSTNAME>/indexserver.ini
- **DATABASE LAYER**: /usr/sap/<SYSTEMNAME>/HDB/custom/config/DV_<dbname>/esserver.ini

The priority of the configuration layers is as follows: DEFAULT < SYSTEM < HOST < DATABASE.

This means that the layer that has the highest priority is the DATABASE layer, followed by the HOST layer, followed by the SYSTEM layer and finally the DEFAULT layer. The configuration with the highest priority is applied to the running environment. If the highest priority level configuration is removed, then the configuration with the next highest priority is applied.

For extended storage, the following are the modifications you can make for configuration parameters:

- You can modify esserver.ini on a systemdb or tenantdb connection.
- You can modify global.ini on a systemdb connection.
- When connected to systemdb, you can modify at the SYSTEM or DATABASE layer for a specific tenant database.
- When connected to a tenant database, you can modify at the SYSTEM layer. Since you are connected to tenant, SYSTEM implies the tenant itself. Specifying DATABASE is unnecessary, and doing so returns an error message.

### Example

You set a parameter new_test_value in the alt_sys_test section of the global.ini file.

```sql
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'SYSTEM') SET ('alt_sys_test', 'new_test_value') = 'test';
```

You check the setting of the parameter in the SAP HANA studio. In the SAP HANA Systems view, right click on the system where you made the change and select Administration then select the Configuration tab. Expand the global.ini configuration file and then the alt_sys_test section. You unset the new_test_value parameter set in the previous step.

```sql
ALTER SYSTEM ALTER CONFIGURATION ('global.ini', 'SYSTEM') UNSET ('alt_sys_test', 'new_test_value');
```

You check the removal of the parameter by clicking refresh on the Configuration page in the SAP HANA studio.

You set the basepath for the data volumes and apply it to the running database.

```sql
ALTER SYSTEM ALTER CONFIGURATION ('esserver.ini', 'SYSTEM') SET ('persistence' 'basepath_datavolumes_es') = '/path/to/data/volumes' WITH RECONFIGURE
```
5.2.7.1 Dynamic Tiering Service Properties (esserver.ini)

The `esserver.ini` file stores extended storage configuration properties.

You can change configuration properties at the system or database layer. Configuration properties changed at the system layer apply to all tenants unless individually overridden at the database level.

Installing dynamic tiering automatically adjusts startup section parameters to new values based on the resources available on the dynamic tiering host.

To change dynamic tiering configuration properties, you need the EXTENDED STORAGE ADMIN system privilege.

### Startup Section Properties

These properties configure startup options for dynamic tiering. The dynamic tiering service must be restarted for changes to these properties to take effect.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Min/Max Value</th>
<th>Default Value/Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalog_cache</td>
<td>Amount of memory initially reserved for caching the dynamic tiering catalog.</td>
<td>0/-</td>
<td>32000000 bytes</td>
</tr>
<tr>
<td>checkpoint_interval</td>
<td>Maximum interval between checkpoints.</td>
<td>0/-</td>
<td>60 minutes</td>
</tr>
<tr>
<td>delta_memory_mb</td>
<td>Amount of memory available to store delta enabled extended tables.</td>
<td>0/-</td>
<td>2048 MB</td>
</tr>
<tr>
<td>es_log_threshold_size</td>
<td>Size (in megabytes) of the log file threshold for point-in-time recovery in dynamic tiering.</td>
<td>1MB/-</td>
<td>Value is based on dbspace size. For databases less than 1TB, the threshold is 10 percent of dbspace size or 2 GB, whichever is greater. For larger databases, the value is 1 percent of dbspace size.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Min/Max Value</td>
<td>Default Value/Remarks</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>heap_memory_mb</td>
<td>Amount of heap memory that dynamic tiering can use. A value of 0 or empty means no limit on heap memory.</td>
<td>0/-</td>
<td>1024 MB</td>
</tr>
<tr>
<td>load_memory_mb</td>
<td>Maximum amount of memory extended storage can request from the operating system for temporary use.</td>
<td>0/-</td>
<td>2048 MB</td>
</tr>
<tr>
<td>main_cache_mb</td>
<td>Amount of memory to be used for caching dynamic tiering database objects.</td>
<td>0/-</td>
<td>1024 MB</td>
</tr>
<tr>
<td>max_concurrent_conections</td>
<td>Maximum number of concurrent connections that the dynamic tiering service accepts.</td>
<td>1 – 1001</td>
<td>50</td>
</tr>
<tr>
<td>max_concurrent_queries</td>
<td>Maximum number of concurrent queries allowed by the server.</td>
<td>0/-</td>
<td>32</td>
</tr>
<tr>
<td>num_partition_buffer_cache</td>
<td>Number of main and temp buffer cache partitions. Must be a power of 2, otherwise value rounded to the nearest power of two to a maximum of 64.</td>
<td>0 – 256</td>
<td>None. Value is determined at runtime based on the number of CPUs, and is not user visible.</td>
</tr>
<tr>
<td>num_threads</td>
<td>Maximum number of threads used for dynamic tiering.</td>
<td>4 – 4096</td>
<td>600</td>
</tr>
<tr>
<td>temporary_cache_mb</td>
<td>Amount of memory to be used as cache for temporary objects during dynamic tiering operations.</td>
<td>0/-</td>
<td>256</td>
</tr>
</tbody>
</table>

### Trace Section Properties

These properties control the size and number of both the esserver and the esserver_console trace files located in the HANA trace directory.

The dynamic tiering service must be restarted for changes to these properties to take effect.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Min/Max Value</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxfiles</td>
<td>Specifies the number of archives of the old message log maintained by the server. Only applies if maxfile-size is not 0.</td>
<td>0 – 64 (inclusive)</td>
<td>10</td>
</tr>
<tr>
<td>maxfilesize</td>
<td>Limits the maximum size of the message log. The value is set in bytes.</td>
<td>0 – 2146435072 bytes (inclusive)</td>
<td>100000000</td>
</tr>
</tbody>
</table>

### Zrlog Section Properties

These properties control request-level logging for dynamic tiering and are typically only used for diagnostic purposes. To enable request-level logging, the property statement_type must be set to a value other than the default (NONE).
The dynamic tiering service must be restarted for changes to these properties to take effect.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>filesize_limit</td>
<td>Create a new log file and rename the original log file when the original log file reaches specified size.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Depends on space available</td>
<td>0</td>
</tr>
<tr>
<td>maxfiles</td>
<td>Specify number of request log file copies to retain. Only takes effect if filesize_limit is also specified.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Depends on space available</td>
<td>0</td>
</tr>
<tr>
<td>statement_type</td>
<td>Enable request logging of operations. Separate multiple values with a comma (,) or plus sign (+).</td>
<td>SQL</td>
<td>HOSTVARS</td>
</tr>
<tr>
<td>tracefile</td>
<td>Redirect request logging information to a file separate from the regular log file.</td>
<td>filename</td>
<td>trace/es_requestlog_${HOST}<em>$ {PORT}</em>$ {COUNT:3}.log</td>
</tr>
</tbody>
</table>

5.2.7.2 Global System Properties (global.ini)

The global.ini file stores global configuration properties for each service in the landscape.

You can add or modify global.ini parameters at the system or database level, but you must make all changes from the SYSTEMDB.

These are the global properties most frequently used for extended storage configuration:

<table>
<thead>
<tr>
<th>Name</th>
<th>Section</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>basepath_database</td>
<td>persistence</td>
<td>Default path for file-based data backups.</td>
<td>File path</td>
<td>($DIR_INSTANCE) /backup/data</td>
</tr>
<tr>
<td>basepath_databa</td>
<td></td>
<td>Default path to the dynamic tiering data directory (data_es). This path must be available to the dynamic tiering host, but need not be available to HANA hosts. Avoid including the default installation directory in the path.</td>
<td>File path</td>
<td>/HANA/data_es/ &lt;SID&gt;</td>
</tr>
<tr>
<td>basepath_logbac</td>
<td></td>
<td>Default path for log backup files. Log backups for dynamic tiering store only the transaction log and metadata related to data modifications to the log backup files.</td>
<td>File path</td>
<td>($DIR_INSTANCE) /backup/log</td>
</tr>
<tr>
<td>basepath_logvol</td>
<td></td>
<td>Default path to the dynamic tiering log directory (log_es). This path must be available to the dynamic tiering host, but need not be</td>
<td>File path</td>
<td>/HANA/log_es/&lt;SID&gt;</td>
</tr>
</tbody>
</table>
available to HANA hosts. Avoid including the default installation directory in the path.

<table>
<thead>
<tr>
<th>Name</th>
<th>Section</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>es_data_backup_buffer_size</td>
<td>backup</td>
<td>Amount of data written or read to (or from) the backup medium during extended storage backup or recovery in a single system call. A smaller value may cause too many system calls, slowing read/write performance on the backup subsystem. A larger value may burden the memory requirements from the dynamic tiering service, and the backup could fail due to lack of memory.</td>
<td>4  – 32</td>
<td>8 MB</td>
</tr>
</tbody>
</table>

You should generally use the SAP HANA cockpit or SAP HANA studio to change parameters instead of editing them directly.

### 5.2.8 ALTER SYSTEM RECONFIGURE SERVICE Statement

[Dynamic Tiering]

Reconfigure a specified service by applying the current configuration parameters.

**Syntax**

```
ALTER SYSTEM RECONFIGURE SERVICE (<service_name>,<hostname>,<port_number>)
```

**Syntax Element**

**service_name**

Specifies the name of the service you wish to reconfigure. See the M_SERVICE_TYPES monitoring view topic for a list of available service types.

```
<service_name> ::= <string_literal>
```

**hostname, port_number**

The hostname and port number where you would like to reconfigure a service.

```
<hostname> ::= <string_literal>
```
Permissions

You must have the EXTENDED STORAGE ADMIN and SERVICE ADMIN system privileges.

Description

You use ALTER SYSTEM RECONFIGURE SERVICE to reconfigure a specified service by applying the current configuration parameters. This command is used after changing multiple configuration parameters with the ALTER CONFIGURATION command without the RECONFIGURE option set. See the ALTER SYSTEM ALTER CONFIGURATION topic.

To reconfigure a specific service specify <hostname> and <port_number> and leave <service_name> empty. To reconfigure all services of a type, specify <service_name> and leave <hostname> and <port_number> empty. To reconfigure all services, leave all parameters empty.

Example

You use the following command to reconfigure all services on the hana.yourcompany.com host using port number 30303:

```
ALTER SYSTEM RECONFIGURE SERVICE ('','hana.yourcompany.com',30303)
```

You use the following command to reconfigure all services of type indexserver:

```
ALTER SYSTEM RECONFIGURE SERVICE ('indexserver','',0)
```

See Also

ALTER SYSTEM ALTER CONFIGURATION Statement [Dynamic Tiering] [page 1312]
5.2.9 ALTER TABLE Statement (Extended Store Table)  
[Dynam ic Tiering]

Changes the definition of an extended store table.

Syntax

Syntax 1 – Alter an Extended Store Table

ALTER TABLE <table_name>  
[<add_column_clause>]  
[<drop_column_clause>]  
[<alter_column_clause>]  
[<add_constraint_clause>]  
[<drop_primary_key_clause>]  
[<drop_constraint_clause>]  
[<owner_to_clause>]

Syntax 2 – Convert an Existing Column Store Table to an Extended Store Table

ALTER TABLE <table_name> <using_extended_storage_clause>

Syntax Elements

table_name

Specifies the identifier of the table to be altered, with optional schema name.

<schema_name> ::= <identifier>
<identifier> ::= [<schema_name>]<identifier>

add_column_clause

Adds one or more columns to the specified table.

<add_column_clause> ::= ADD ( {<column_definition> [COMMENT <string_LITERAL>] } [, ... ]

drop_column_clause

Removes one or more columns from the specified table.

<drop_column_clause> ::= DROP ( <column_name>[,...] )

alter_column_clause

Alters one or more column definitions.

<alter_column_clause> ::= ALTER ( {<column_definition> [COMMENT <string_LITERAL>] } [,... ]

[Dynamic Tiering]
Restrictions:

- For column tables, changing data type from a string to number (and back) is supported as long as the data fits into the target column size.
- For row tables, only increasing the size of VARCHAR and NVARCHAR type column is allowed. Other data type changes are not allowed.
- ALTER does not currently follow data type conversion rules.
- Adding NOT NULL constraint to an existing column is allowed if either of the following cases are true:
  - The table is empty.
  - The default value is specified when the table contains data.
  - The table does not contain a NULL-value in that column.
- You can only alter the comment for a column while making another change to the column (such as changing the data type).

`column_definition`

Defines a table column.

```sql
<column_definition> ::= <column_name> {<data_type> | <lob_data_type>} [<default_value_clause>]
```

`column_name`

Specifies the table column name.

```sql
<column_name> ::= <identifier>
```

`data_type`

The available data types.

```sql
<data_type> ::= DATE | TIME | SECONDDATE | TIMESTAMP | TINYINT | SMALLINT | INTEGER | BIGINT | SMALLDECIMAL | DECIMAL | REAL | DOUBLE | VARCHAR | NVARCHAR | ALPHANUM | SHORTTEXT | VARBINARY | TEXT | BINTEXT
```

```sql
<lob_data_type> ::= <lob_type_name>
```

```sql
<lob_type_name> ::= BLOB | CLOB | NCLOB
```

`default_value_clause`
Specifies a value to be assigned to the column if an INSERT statement does not provide a value for the column.

```plaintext
<default_value_clause> ::= DEFAULT <default_value_exp>
```

```plaintext
<default_value_exp> ::= NULL
| <string_literal>
| <signed_numeric_literal>
| <unsigned_numeric_literal>
| <datetime_value_function>
```

```plaintext
<datetime_value_function> ::= CURRENT_DATE
  | CURRENT_TIME
  | CURRENT_TIMESTAMP
```

column_constraint

The column constraint rules.

```plaintext
<column_constraint> ::= NULL
  | NOT NULL
  | <unique_specification>
```

NULL

If NULL is specified it is not considered a constraint, it represents that a column that may contain a null value. The default is NULL.

NOT NULL

The NOT NULL constraint prohibits a column value from being NULL.

unique_specification

Specifies a unique constraint.

```plaintext
<unique_specification> ::= UNIQUE [<unique_tree_type_index>]
  | PRIMARY KEY [<unique_tree_type_index>]
```

If the index type is omitted, the SAP HANA database chooses the appropriate index by considering the column data type. If the index type is not specified, the SAP HANA database automatically selects an index type as follows:

<table>
<thead>
<tr>
<th>Index type</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPBTREE</td>
<td>• character string types</td>
</tr>
<tr>
<td></td>
<td>• binary string types</td>
</tr>
<tr>
<td></td>
<td>• decimal types</td>
</tr>
<tr>
<td></td>
<td>• when the constraint is a composite key</td>
</tr>
<tr>
<td></td>
<td>• when the constraint is a non-unique constraint</td>
</tr>
<tr>
<td>BTREE</td>
<td>All other cases than specified for CPBTREE</td>
</tr>
</tbody>
</table>

Specifies a column as a unique key. A composite unique key enables the specification of multiple columns as a unique key. With a unique constraint, multiple rows cannot have the same value in the same column.
Specifies a column as a primary key. A primary key constraint is a combination of a NOT NULL constraint and a UNIQUE constraint. It prohibits multiple rows from having the same value in the same column.

**unique_tree_type_index**

Specifies the index type.

```
<unique_tree_type_index> ::= BTREE | CPBTREE;
```

Specifies a B+-tree index. B+-tree is a tree that maintains sorted data, which performs efficient insertion, deletion and search of records.

CPBTREE specifies a CPB+-tree index. CPB+-tree stands for Compressed Prefix B+-Tree, which is based on pkB-tree. CPB+-tree is a very small index because it uses 'partial key' that is only part of full key in index nodes. CPB+-tree shows better performance than B+-Tree for larger keys.

**add_constraint_clause**

Adds a table constraint.

```
<add_constraint_clause> ::= ADD [CONSTRAINT <constraint_name>] <table_constraint>
<constraint_name> ::= <identifier>
<table_constraint> ::= <unique_constraint_definition>
```

**table_constraint**

The table constraint can be either a unique constraint.

```
<table_constraint> ::= <unique_constraint_definition>
```

**unique_constraint_definition**

The unique specification.

```
<unique_constraint_definition> ::= <unique_specification> (<unique_column_name_list>)
```

**unique_column_name_list**

Specifies the unique column name list, which can have one or more column names.

```
<unique_column_name_list> ::= <unique_column_name>[, <unique_column_name>...] <unique_column_name> ::= <identifier>
```

**drop_primary_key_clause**

Drops the primary key constraint.

```
<drop_primary_key_clause> ::= DROP PRIMARY KEY
```

**drop_constraint_clause**

Drops a unique or referential constraint.

```
<drop_constraint_clause> ::= DROP CONSTRAINT <constraint_name>
<constraint_name> ::= <identifier>
```

**using_extended_storage_clause**
Modifies table to be an extended store table.

<using_extended_storage_clause> ::= [NOT] USING EXTENDED STORAGE [(ENABLE | DISABLE) DELTA]

\*\* Note \*\*

- Extended storage must already exist.
- DBO, SYS, and PUBLIC are reserved words within extended storage. Schemas with these names cannot have extended store tables.
- DELTA must be enabled on extended storage to enable at enable on extended table.
- DELTA not supported for data types BLOB, CLOB, and NCLOB.

\*\* NOT USING EXTENDED STORAGE \*\*

Converts an extended table to a column store table.

\*\* USING EXTENDED STORAGE \*\*

Converts a column store table to an extended table.

\*\* ENABLE DELTA \*\*

Allows row level versioning on an extended table. DELTA facilitates faster inserts.

\*\* DISABLE DELTA \*\*

Disables row level versioning on an extended table.

owner_to_clause

Specifies a new owner for the table.

<owner_to_clause> ::= OWNER TO <user_name>

The new owner becomes the grantor for all of the privileges on the table that were granted by the old owner. Also, the old owner is granted all privileges for the table by the new owner so that they can continue running jobs on the table.

Permissions

You must have the EXTENDED STORAGE ADMIN privilege.

Description

The ALTER TABLE (extended store table) statement changes the definition of an extended store table.
Examples

To convert table t1 to an extended store table with DELTA enabled.

```
ALTER TABLE t1 USING EXTENDED STORAGE ENABLE DELTA;
```

To convert extended store table t1 to be an SAP HANA table.

```
ALTER TABLE t2 NOT USING EXTENDED STORAGE
```

To enable delta on extended store table t3.

```
ALTER TABLE t3 USING EXTENDED STORAGE ENABLE DELTA
```

To add a column C to extended store table t4.

```
ALTER TABLE t4 ADD (C INT)
```

To drop column C on extended store table t4.

```
ALTER TABLE t4 DROP (C)
```

To add column constraint UK on column B on extended store table t4.

```
ALTER TABLE t4 ADD CONSTRAINT UK UNIQUE (B)
```

To drop column constraint UK on extended store table t4.

```
ALTER TABLE t4 DROP CONSTRAINT UK
```

To rename extended store table t4 to t5.

```
RENAME TABLE t4 TO t5
```

To delete extended store table t5.

```
DROP TABLE t5
```

Related Information

- TABLES System View [page 1669]
- TABLE_COLUMNS System View [page 1675]
- M_ES_TABLES System View [Dynamic Tiering] [page 2305]
- TABLES System View [page 1669]
- TABLE_COLUMNS System View [page 1675]
- M_ES_TABLES System View [Dynamic Tiering] [page 2305]
5.2.10 ALTER TABLE Statement (Multistore Table) [Dynamic Tiering]

Alters the definition of a multistore table.

Syntax

```
ALTER TABLE <table_name>
[ <add_column_clause> ]
[ <drop_column_clause> ]
[ <alter_column_clause> ]
[ <add_constraint_clause> ]
[ <drop_constraint_clause> ]
[ <move_column_store_partition_clause> ]
[ <persistent_merge_clause> ]
[ <delta_log_clause> ]
[ <auto_merge_clause> ]
[ <unload_priority_clause> ]
[ <set_group_clause> ]
[ <unset_group_clause> ]
[ <reclaim_data_space_clause> ]
[ <non-heterogeneous_partition_clauses> ]
[ <heterogeneous_partition_clauses> ]
[ <enable_delta_for_extended_storage> ]
[ <owner_to_clause> ]
```

Syntax Elements

table_name

Specifies the identifier of the table to be altered, with optional schema name.

```
<table_name> ::= [<schema_name>.[]<identifier>  <schema_name> ::= <identifier>
```

add_column_clause

Adds one or more columns to the specified table.

```
<add_column_clause> ::= ADD ( {<column_definition> [column_constraint_short]} [COMMENT <string_literal>] ) [, ... ]
```

column_definition

Defines a table column.

```
<column_definition> ::= <column_name> <data_type> [lob_data_type] [default_value_clause]
```

column_name
The table column name.

\[<column\_name> ::= <identifier>\]

data_type

The available data types.

\[<data\_type> ::= DATE \]
\[TIME \]
\[SECONDDATE \]
\[TINYINT \]
\[SMALLINT \]
\[INTEGER \]
\[BIGINT \]
\[DECIMAL [ , ] \]
\[REAL \]
\[DOUBLE \]
\[VARCHAR [ <unsigned\_integer> ] \]
\[NVARCHAR [ <unsigned\_integer> ] \]
\[VARBINARY [ <unsigned\_integer> ] \]

lob_data_type

Specifies the column constraint rules.

\[<lob\_data\_type> ::= BLOB \]
\[CLOB \]
\[NCLOB \]

column_constraint_short

\[<column\_constraint\_short> ::= NULL | NOT NULL\]

NULL

If NULL is specified it is not considered a constraint, it represents that a column that may contain a null value. The default is NULL.

NOT NULL

The NOT NULL constraint prohibits a column value from being NULL.

default_value_clause

Specifies a value to be assigned to the column if an INSERT statement does not provide a value for the column.

\[<default\_value\_clause> ::= DEFAULT <default\_value\_exp>\]

\[<default\_value\_exp> ::= NULL \]
\[<string\_literal> \]
\[<signed\_numeric\_literal> \]
\[<unsigned\_numeric\_literal> \]
\[<datetime\_value\_function> \]

\[<datetime\_value\_function> ::= CURRENT\_DATE \]
\[CURRENT\_TIME \]
\[CURRENT\_TIMESTAMP \]

drop_column_clause

SAP HANA SQL Reference Guide for SAP HANA Platform
SQL Reference for Additional SAP HANA Contexts
Removes columns from the specified table. Columns must be dropped one at a time.

\[
\text{\textless drop_column_clause\textgreater} ::= \text{DROP} \ (\text{\textless column_name\textgreater})
\]

**alter_column_clause**

Alters a column definition.

\[
\text{\textless alter_column_clause\textgreater} ::= \text{ALTER} \ (\text{\textless column_definition\textgreater} \ [\text{\textless column_constraint\textgreater})
\]

For more information on column definition, see column_definition [page 1326].

**Restrictions:**

- To prevent data loss, you can only increase the size of a column data type. For example, changing from nvarchar(20) to nvarchar(10) or from integer to tinyint raises an error.
- ALTER does not currently follow data type conversion rules. Adding the NOT NULL constraint to an existing column is allowed if either of the following cases are true:
  - The table is empty.
  - The table does not contain a NULL-value in that column.
- Primary keys (or secondary indexes) with multiple columns are not allowed in these conditions:
  - When the total key length (sum of all column sizes) is greater than 5300 bytes.
  - When the column types include NVARCHAR and at least one column has a length greater than 1765 bytes.
- Columns with data type TEXT are not supported.
- You cannot drop or modify a partition key column.
- Multistore tables only support adding, dropping, or modifying columns one column at a time.

**column_constraint**

Specifies all column constraints.

\[
\text{\textless column_constraint\textgreater} ::= \text{NULL} \ | \text{NOT NULL} \ | \text{\textless unique_specification\textgreater}
\]

**NULL**

If specified, it is not considered a constraint, but represents a column that may contain a null value. The default is NULL.

**NOT NULL**

Prohibits a column value from being NULL.

**unique_specification**

 Specifies unique constraints.

\[
\text{\textless unique_specification\textgreater} ::= \text{UNIQUE} \ [\text{\textless unique_tree_type_index\textgreater}] \ | \text{PRIMARY KEY} \ [\text{\textless unique_tree_type_index\textgreater}]
\]

If you omit the index type, the SAP HANA database chooses the appropriate index automatically based on data type or constraint:

- CPBTREE:
• Character string types
• Binary string types
• Decimal types
• When the constraint is a composite key
• When the constraint is a non-unique constraint

• BTREE: All other cases than specified for CPBTREE

**PRIMARY KEY**

A primary key constraint is a combination of a NOT NULL constraint and a UNIQUE constraint. It prohibits multiple rows from having the same value in the same column.

**unique_tree_type_index**

Specifies the index type.

```sql
<unique_tree_type_index> ::= BTREE | CPBTREE;
```

**BTREE**

BTREE specifies a B+ tree index. B+ tree is a tree that maintains sorted data, which performs efficient insertion, deletion and search of records.

**CPBTREE**

CPBTREE specifies a CPB+ tree index. CPB+ tree stands for Compressed Prefix B+ tree, which is based on pkB tree. CPB+ tree is a very small index because it uses partial key that is only part of full key in index nodes. CPB+ tree shows better performance than B+ tree for larger keys.

**add_constraint_clause**

Adds a table constraint.

```sql
<add_constraint_clause> ::= ADD [CONSTRAINT <constraint_name>] <table_constraint> [FOR {DEFAULT|EXTENDED} STORAGE]
```

**<constraint_name> ::= <identifier>**

**table_constraint**

Only unique constraints are supported on a multistore table.

```sql
<table_constraint> ::= <unique_constraint_definition>
```

**unique_constraint_definition**

The unique specification.

```sql
<unique_constraint_definition> ::= <unique_specification> ....(<unique_column_name_list>)
```

**unique_constraint_definition**

Specifies the unique column name list, which can have one or more column names.

```sql
<unique_column_name_list> ::= <unique_column_name> [{,<unique_column_name>}...]
```

**unique_column_name**
A column name identifier.

<unique_column_name> ::= <identifier>

FOR {DEFAULT|EXTENDED} STORAGE]

Specifies the location on a multistore table where the constraint is internally created. You may create the constraint as a partial index on either the default or extended storage. If you don’t specify storage, the constraint is created on both the default and extended storage.

drop_primary_key_clause

Drops the primary key constraint.

<drop_primary_key_clause> ::= DROP PRIMARY KEY [FOR {DEFAULT|EXTENDED} STORAGE]

If you drop a primary key on a partial index for a multistore table, you may drop it from the store where it resides. For example, FOR DEFAULT STORAGE drops the default store part of the primary key. If you don’t specify storage, the primary key is totally dropped.

drop_constraint_clause

Drops a unique constraint.

<drop_constraint_clause> ::= DROP CONSTRAINT<constraint_name> [FOR {DEFAULT|EXTENDED} STORAGE]  
<constraint_name> ::= <identifier>

If you drop a constraint created as a partial index, you may drop it from the store where it resides and convert it into a partial index on the other store. For example, FOR DEFAULT STORAGE drops the constraint from the default store and converts it into a partial index on the extended store. If you don’t specify storage, the constraint is totally dropped.

move_column_store_partition_clause

Moves a table or partition to another indexserver location in a distributed environment

i Note

The <move_clause> parameter is only supported on column store partitions.

<move_column_store_partition_clause> ::= MOVE [ PARTITION <partition_number> ]  
| TO <indexserver_host_port> [PHYSICAL]  
| MOVE [PARTITION <partition_number>] PHYSICAL

partition_number

Specifies the partition to be moved. You see an error if you try to move a partitioned table without specifying <partition_number>.

<partition_number> ::= <unsigned_integer>

indexserver_host_port

The internal indexserver host name and port number where the table is to be moved.

<indexserver_host_port> ::= 'host_name:port_number'

PHYSICAL
The PHYSICAL keyword is only for column store tables. Row store tables are always moved immediately.

**persistent_merge_clause**

Enables or disables persistent merging.

```
<persistent_merge_clause> ::= { ENABLE | DISABLE } PERSISTENT MERGE
```

- (Default) When enabled, the merge-manager uses persistent merges for the given column store table.
- When disabled, the merge-manager uses main-memory merges instead of persistent merges for the given column store table.

**delta_log_clause**

Enables or disables the redo log for the table.

```
<delta_log_clause> ::= { ENABLE | DISABLE } DELTA LOG
```

Use this clause only during initial load.

If you enable logging, perform a savepoint afterward to make sure that all data persists. Next, perform a data backup so that you can recover the data.

If you disable logging, log entries do not persist for this table and savepoints only write changes to the data store. This can cause loss of committed transactions if the indexserver terminates, in which case you have to truncate this table and insert all data again.

**auto_merge_clause**

Enables or disables automatic delta merge on the specified table.

The `<auto_merge_clause>` parameter is supported on column store partitions.

If not specified, the table move will create a link inside. Specifies that a column store tables persistence storage is moved immediately to the target host. If the PHYSICAL option parameter is supported on the new host persistence pointing to the old host persistence. The link will be removed on the next merge or column store partitions. upon execution of another move operation not using the TO is not specified the table move will create a link inside the new host persistence pointing to the old host.

```
<auto_merge_clause> ::= { ENABLE | DISABLE } AUTOMERGE
```

**unload_priority_clause**

Specifies that priority of table to be unloaded from memory.

```
<unload_priority_clause> ::= UNLOAD PRIORITY <unload_priority>
```

Notes:
- The `<persistent_mergeClause>` parameter is supported on column store partitions.
- The `<delta_logClause>` parameter is supported on column store partitions.
- The `<unload_priorityClause>` parameter is supported on column store partitions.
set_group_clause
Sets the group type, subtype, and name.

<group_clause> ::= SET <group> [ <group> ... ]
<group> ::= GROUP TYPE <identifier> | GROUP SUBTYPE <identifier> | GROUP NAME <identifier>

Grouping applies only to column store partitions within a multistore table.

unset_group_clause
Unsets the group type, subtype, and name.

<unset_group_clause> ::= <UNSET GROUP>

reclaim_data_space_clause
Reclaims data space of the specified table.

Note
The reclaim_data_space_clause parameter is a row-only clause not supported on column store tables.

<reclaim_data_space_clause> ::= RECLAIM DATA SPACE

This statement defragments your database by restructuring table data space. However, this does not always guarantee that restructured data space uses less memory than before. In some optimized cases, restructured data space uses much memory.

non-heterogeneous_partition_clauses
Alter operations you can perform on a non-heterogeneous partitioned multistore table.

[ <add_range_clause>] [ <drop_range_clause>] [ <redefine_range_clause>] [ <alter_range_partition_attributes_clause>] [ <move_range_clause>]

You cannot convert a partitioned column store or extended store table directly to a non-heterogeneous partitioned multistore table. It requires multiple ALTER TABLE commands. For more information, see the Manage a Multistore Table section in the SAP HANA Dynamic Tiering Administration Guide.

add_range_clause
Adds a range partition to an existing multistore table with a non-heterogeneous partitioning scheme.

<add_range_clause> ::= ADD PARTITION { <column_name> } USING {DEFAULT | EXTENDED } STORAGE { <range_values> }

Only one range can be added at a time. The USING {DEFAULT | EXTENDED } STORAGE parameter is required when adding a range to an extended storage partition, but optional for a default
storage partition. This clause is not supported if you want to add a range to an extended storage partition and PARTITION OTHERS exists in default storage. For this scenario, you must use both the `<redefine_range_clause>` and `<move_range_clause>` clauses. For more information, see the Manage a Multistore Table section in the SAP HANA Dynamic Tiering Administration Guide.

**column_name** Specifies the name of the existing partitioning column the range refers to.

**range_values**

Specifies the range of the partition.

```plaintext
<range_values> ::= <min_value> <= VALUES <max_value>
VALUES = <target_value>
OTHERS
```

**min_value**

The lower value of a partition specifier. Value cannot be negative.

```plaintext
<min_value> ::= <string_literal> | <numeric_literal>
```

**max_value**

The upper value of a partition specifier. Value cannot be negative.

```plaintext
<max_value> ::= <string_literal> | <numeric_literal>
```

**target_value**

The target value of a single partition specifier. Value cannot be negative.

```plaintext
<target_value> ::= <string_literal> | <numeric_literal>
```

**OTHERS**

Specifies that all other values that are not covered by the partition range values will be gathered into one partition. OTHERS can only be defined within a default store partition.

**drop_range_partition_clause**

Specifies the range of the partition to drop. Only a single range can be dropped at a time.

```plaintext
<drop_range_partition_clause> ::= DROP PARTITION ( <column_name> )
<range_values>
```

**redefine_range_clause**

Modifies the ranges of an existing non-heterogeneous partition.

```plaintext
<redefine_range_clause> ::= PARTITION BY RANGE (<column_name>)
( USING DEFAULT STORAGE ( <range_values> [ , <range_values> ] )
USING EXTENDED STORAGE ( <range_values> [ , <range_values> ] )
)
```

You can add new ranges or split or merge existing ranges as long as the new ranges provide a valid partition for all existing data within the table, and no data moves between stores as a result of the redefined ranges. If the redefined ranges results in existing data falling outside of the ranges, or moving to a different store the alter command fails. If a partition is empty, and it is not included in the redefined ranges, then it is dropped.

For example, range 10-20 resides in default storage and range 20-30 resides in extended storage. The values 10 through 17 are inserted. If range 10-20 is redefined as 10-15, then the command fails because...
the new range excludes the existing values of 16 and 17. However, if the range is redefined to 10-18, then the command succeeds because all of the existing data is still included in the redefined range. If the range in default storage is redefined to 10-15 and in extended storage to 15-30, then the command fails because even though all the data is still included within the ranges, the redefined ranges would result in the values 16 and 17 moving to extended storage.

**alter_range_partition_attributes_clause**

Defines attributes on the partition.

```plaintext
<alter_range_partition_attributes_clause> ::= FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS { ON | OFF }
FOR DEFAULT STORAGE {ALL | NON CURRENT } PARTITIONS { PAGE | COLUMN } LOADABLE
```

**move_range_clause**

Moves an existing partition range between stores. Only one range can be moved at a time.

```plaintext
<move_range_clause> ::= ALTER PARTITION ( <column_name> ) [ USING DEFAULT STORAGE | USING EXTENDED STORAGE ] ( <range_values> )
```

**heterogeneous_partition_clauses**

Alter operations you can perform on a heterogeneous partitioned multistore table.

```plaintext
[ <add_heterogeneous_range_clauses>]
[ <drop_heterogeneous_partition_clauses>]
[ <move_heterogeneous_partition_clause>]
[ <redefine_heterogeneous_range_clause>]
[ <alter_heterogeneous_partition_primary_key_clause> ]
```

You cannot convert a partitioned column store or extended store table directly to a heterogeneous partitioned multistore table. It requires multiple ALTER TABLE commands. For more information, see the Manage a Multistore Table section in the SAP HANA Dynamic Tiering Administration Guide.

**add_heterogeneous_range_clauses** Alter operations you can perform to add new first- or second-level range partitions to an existing multistore table.

**add_het_range_partition_clause**

Adds a range partition to either store. You must add at least one subpartition when initially adding the partition to be able to add additional subpartitions later.

```plaintext
<add_het_range_partition_clause> ::= ADD PARTITION RANGE (<column_name>)
( { PARTITION <het_range_spec> <het_location>[, PARTITION}
<het_range_spec> <het_location> ...] }
SUBPARTITION BY RANGE (<column_name>) ( PARTITION
<het_range_spec> [, PARTITION <het_range_spec> ... ] )
| <het_range_spec>::= <het_range_values> [INSERT {OFF | ON} ]

column_name Specifies the name of the existing partitioning column the range is being added to.

**het_range_values**

Specifies the range of the partition.

```plaintext
<het_range_values> ::= <min_value> <= VALUES < <max_value>
VALUES = <target_value>
OTHERS
```

**min_value**
The lower value of a partition specifier. Value cannot be negative.

\[
<\text{min\_value}> ::= <\text{string\_literal}> | <\text{numeric\_literal}>
\]

max_value

The upper value of a partition specifier. Value cannot be negative.

\[
<\text{max\_value}> ::= <\text{string\_literal}> | <\text{numeric\_literal}>
\]

target_value

The target value of a single partition specifier. Value cannot be negative.

\[
<\text{target\_value}> ::= <\text{string\_literal}> | <\text{numeric\_literal}>
\]

OTHERS

Specifies that all other values that are not covered by the partition specification will be gathered into one partition. Can only be added to a default store partition.

**INSERT { ON | OFF }**

(Applies to heterogeneous range partitions only) Specifies if INSERT statements are allowed on a partition. The INSERT parameter can be defined on a first or second level partition. If defined at the first level, it applies to all second level partitions within the first level partition. If defined at both the first and second level, any value at the second level overrides the first level value. If not specified, default is ON.

het_location

Specifies where the partition resides. If a location is not specified, the partition resides in default storage.

\[
<\text{location}> ::= \text{AT LOCATION}
\]

\[
\{'<\text{HANA\_host}>::<\text{HANA\_port}>' | '<\text{ES\_host}>::<\text{ES\_port}>'\}
\]

**add_het_range_subpartition_clause**

Adds second-level range partitions to existing first-level range partitions.

\[
<\text{add\_het\_range\_subpartition\_clause}> ::= 
\text{ALTER PARTITION RANGE } <\text{column\_name}> \{ ( \text{PARTITION } <\text{het\_range\_values}> [, \text{PARTITION } <\text{het\_range\_values}> \ldots] ) \}
\text{ADD PARTITION RANGE } <\text{column\_name}> \{ ( \text{PARTITION } <\text{het\_range\_values}> [, \text{PARTITION } <\text{het\_range\_values}> \ldots] ) \}
\]

The existing first-level ranges specified can be in either store. When multiple first-level partitions are specified, the subpartitions are added to each first-level partition. You can add the same subpartition range to multiple first-level partitions regardless of their store at the same time. For example, in a single ALTER statement, you could add the subpartition 500-1000 to first level-partitions 10-20 and 20-30, in default storage, and partitions 30-40 in extended storage.

**drop_heterogeneous_partition_clauses**

Drops first- or second-level range partitions. Dropping a first-level range partition drops all associated subpartitions and deletes any data contained within the partition or associated subpartitions.

**drop_het_range_partition_clause**
Specifies the first level range being dropped.

```plaintext
drop_het_range_partition_clause ::=     DROP PARTITION RANGE ( <column_name> )         ( ( PARTITION <range_values> [, PARTITION <range_values> ...] ) )
```

drop_het_range_subpartition_clause

Drop a subpartition from a first level range partition. Dropping the last subpartition deletes the first-level partition as well.

```plaintext
<drop_heterogeneous_subpartition_clause> ::=     ALTER PARTITION RANGE (<column_name >)         ( (PARTITION <range_values> [ , PARTITION <range_values> ... ] ) )          DROP PARTITION RANGE (<column_name>) ( ( PARTITION <range_values> [ , PARTITION <range_values> ... ] ) )
```

move_heterogeneous_partition_clause

Move a first-level partition on a heterogeneous range multistore table to a new store. Any second level partitions associated with the partition being moved are also moved. The partition being moved must be a standalone defined range. To move part of an existing range, alter the partition range to isolate the part and then move it. Moving a partition requires the `<partition_number>` of the partition. To locate this value, execute:

```sql
SELECT * FROM table_partitions
```

The `<partition_number>` is the value in the LEVEL_1_PARTITION column on the row of the range to move.

```plaintext
<move_heterogeneous_partition_clause> ::= MOVE PARTITION <partition_number> TO <move_location>
```

move_location

Specifies the location the partition is being moved to. The new location can be within the default store on a different indexserver or the extended store on the dynamic tiering server.

```plaintext
<move_location>::=      '<hana_host>:<port>'      | '<es_server_host>:<port>'
```

redefine_heterogeneous_range_clause

Modifies the existing ranges and INSERT property of first- and second-level partitions. The new ranges must provide a valid partition for all existing data within the table. If a partition or subpartition is empty, and it is not included in the redefined ranges, it is dropped. Data cannot move between stores as a result of the redefined ranges.

```plaintext
<redefine_heterogeneous_range_clause> ::= PARTITION BY RANGE (<column_name>)      ( ( PARTITION <range_values> [, PARTITION <range_values> ...]          [ SUBPARTITION BY RANGE (<column_name>) ( PARTITION <range_values> [ ,PARTITION <range_values> ... ] ) ],       ( PARTITION <range_values> [,PARTITION <range_values> ... ] [ SUBPARTITION BY RANGE (PARTITION <column_name>) ( PARTITION <range_values> [ ,<partition_subrange> ... ] ) ]     ) [ PRIMARY KEY UPDATE { ON | OFF } ]
```

alter_heterogeneous_partition_primary_key_clause
Specifies if UPDATE statements are allowed on primary key columns.

```sql
<alter_heterogeneous_partition_primary_key_clause> ::= PRIMARY KEY UPDATE {ON | OFF}
```

**enable_delta_for_extended_storage**

Specifies whether extended storage is to use row-level versioning. This clause applies only to extended storage partitions in a multistore table.

```sql
<enable_delta_for_extended_storage> ::= FOR EXTENDED STORAGE [ ENABLE | DISABLE ] DELTA
```

**owner_to_clause**

Specifies a new owner for the table.

```sql
<owner_to_clause> ::= OWNER TO <user_name>
```

The new owner becomes the grantor for all of the privileges on the table that were granted by the old owner. Also, the old owner is granted all privileges for the table by the new owner so that they can continue running jobs on the table.

**Permissions**

You must have the EXTENDED STORAGE ADMIN privilege.

**Description**

The ALTER TABLE (multistore table) statement changes the definition of a multistore table.

When managing partitions on a multistore table, refer to the section Manage a Multistore Table in the SAP HANA Dynamic Tiering Administration Guide for details and specific examples on supported clause behaviors.

**Examples**

The following examples are based on multistore table t2:

```sql
CREATE TABLE t2 (a int primary key, b int) 
PARTITION BY HASH (a) PARTITIONS 2, 
RANGE (b) 
(USING DEFAULT STORAGE (PARTITION 1 <= VALUES < 10 ) 
USING EXTENDED STORAGE (PARTITION 10 <= values <20 ));
```

This example adds column c with a data type in integer to the existing table t2.

```sql
ALTER TABLE t2 ADD (c int);
```
This example changes the data type of column c from integer to date on table t2.

```sql
ALTER TABLE t2 ALTER (c DATE);
```

This example removes column c from table t2.

```sql
ALTER TABLE t2 DROP (c);
```

This example adds the unique constraint UK to multistore table t2.

```sql
ALTER TABLE t2 ADD CONSTRAINT UK UNIQUE (B);
```

This example drops contraint UK from table t2.

```sql
ALTER TABLE t2 DROP CONSTRAINT UK;
```

This example sets a group type named test on table t2.

```sql
ALTER TABLE t2 SET GROUP TYPE "test";
```

This example removes the group from table t2.

```sql
ALTER TABLE t2 UNSET GROUP;
```

This example changes the owner of table t2 to user1.

```sql
ALTER TABLE t2 OWNER to user1;
```

**Partitioning**

This example enables delta or disables on extended storage on table t2. It syntax works on both non- and heterogeneous range multistore tables.

```sql
ALTER TABLE t2 FOR EXTENDED STORAGE ENABLE DELTA;

ALTER TABLE t2 FOR EXTENDED STORAGE DISABLE DELTA;
```

**Non-heterogeneous Partitioning**

The following examples are based on this non-heterogeneous partitioned multistore table:

```sql
CREATE COLUMN TABLE t1 (a INT, b INT)
    PARTITION BY RANGE (b)
    ( USING DEFAULT STORAGE ( PARTITION 10 <= VALUES < 15, PARTITION 15 <= VALUES < 30 )
    USING EXTENDED STORAGE ( PARTITION 30 <= VALUES < 40, PARTITION 40 <= VALUES < 50 ) );
```

This example adds partition 5-10 to default storage.

```sql
ALTER TABLE t1 ADD PARTITION (b)
    USING DEFAULT STORAGE (5 <= VALUES < 10);
```

This example adds partition 60-70 to extended storage.

```sql
ALTER TABLE t1 ADD PARTITION (b)
    USING EXTENDED STORAGE (60 <= VALUES < 70);
```
This example redefines the existing default storage partitions on a range partitioned table. It splits range 15-30 into 15-20 and 20-30. It also adds the OTHERS partition to default storage:

```sql
ALTER TABLE t1 PARTITION BY RANGE (b)     ( USING DEFAULT STORAGE ( PARTITION 10 <= VALUES < 15, PARTITION 15 <= VALUES < 20, PARTITION 20 <= VALUES < 30, PARTITION OTHERS)     USING EXTENDED STORAGE ( PARTITION 30 <= VALUES < 40, PARTITION 40 <= VALUES < 50) ) ;
```

This example drops partition range 10-15.

```sql
ALTER TABLE t1 DROP PARTITION (b) 10 <= VALUES < 15;
```

This example moves partition range 15-20 from default storage to extended storage.

```sql
ALTER TABLE t1 USING EXTENDED STORAGE 15 <= VALUES < 20;
```

**Heterogeneous Range Partitioning**

The following examples are based on this heterogeneous range partitioned table:

```sql
CREATE COLUMN TABLE t2 (a INT, b INT)      PARTITION BY RANGE(a)     ( (PARTITION 5 <= VALUES < 10, PARTITION 10 <= VALUES < 15, PARTITION 15 <= VALUES < 20 USING DEFAULT STORAGE)     SUBPARTITION BY RANGE (b) (PARTITION VALUES = 100), (PARTITION 50 <= VALUES < 60 USING EXTENDED STORAGE)     SUBPARTITION BY RANGE (b) (PARTITION VALUES = 200) ) ;
```

This example redefines two existing extended storage partition range, 5-10 and 10-15 into a single partition range.

```sql
ALTER TABLE t2 PARTITION BY RANGE (a)     ( (PARTITION 5 <= VALUES < 15, PARTITION 15 <= VALUES < 20 )     SUBPARTITION BY RANGE (b) (PARTITION VALUES = 100), (PARTITION 50 <= VALUES < 60)     SUBPARTITION BY RANGE (b) (PARTITION VALUES = 200) ) ;
```

This example adds first-level range partition 60-70 and subpartition 100-200 to extended storage.

```sql
ALTER TABLE t2 ADD PARTITION RANGE (a)     ( { PARTITION 60 <= VALUES < 70 AT LOCATION 'myDTserver:30112' }     SUBPARTITION BY RANGE (b) (PARTITION 100 <= VALUES < 200) ) ;
```

This example adds subpartition value=200 to range partition 15-20 in default storage.

```sql
ALTER TABLE t2 ALTER PARTITION RANGE (a)     ( { PARTITION 15 <= VALUES < 20 } )     ADD PARTITION RANGE (b) ( { PARTITION VALUES = 200 } ) ;
```

This example adds the OTHERS partition to default storage.

```sql
ALTER TABLE t2 ADD PARTITION RANGE (a)     ( { PARTITION OTHERS } ) ;
```

This example drops only subpartition 200 from range 15-20. Range 15-20 remains.

```sql
ALTER TABLE t2 DROP PARTITION RANGE (a)
```
This example drops partition 60-70 from extended storage. This also drops its subpartition 100-200.

```
ALTER TABLE t2 DROP PARTITION RANGE (a)
  ( ( PARTITION 60 <= VALUES < 70 ) );
```

This example moves partition 5-15, which is partition 1 to extended storage.

```
ALTER TABLE T2 MOVE PARTITION 1 TO 'myDTserver:30112';
```

This example adds the ability to run UPDATE statements on the primary key in the multistore table.

```
ALTER TABLE T2 PRIMARY KEY UPDATE ON;
```

**Related Information**

- TABLES System View [page 1669]
- TABLE_COLUMNS System View [page 1675]
- PARTITIONED_TABLES System View [page 1606]
- TABLE_PARTITIONS System View [page 1681]
- M_ES_TABLES System View [Dynamic Tiering] [page 2305]
- CREATE TABLE Statement (Multistore Table) [Dynamic Tiering] [page 1379]
- TABLES System View [page 1669]
- TABLE_PARTITIONS System View [page 1681]
- TABLE_COLUMNS System View [page 1675]
- M_ES_TABLES System View [Dynamic Tiering] [page 2305]

**Table Partitioning**

### 5.2.11 CALL CHECK_CATALOG Statement (Multistore Table) [Dynamic Tiering]

Performs consistency checks of metadata in both multistore and extended store tables, based on the actions you specify as arguments to the procedure call.

**Syntax**

```
CALL CHECK_CATALOG ('<action>', '<schema_name>', '<object_name>',
  '<catalog_object_type>')
```
Syntax Elements

**action** The type of consistency check to perform.

```latex
<action> ::= CHECK_ES_METADATA | REPAIR_ES_METADATA | CHECK | CHECK_VALUE_DOMAIN | CHECK_OBJECT_REFERENTIAL_INTEGRITY
```

**CHECK_ES_METADATA**
Checks the metadata of a multistore or extended store table for inconsistencies such as missing tables, columns, and indexes.

**REPAIR_ES_METADATA**
Fixes metadata inconsistencies by deleting orphaned objects found by running CHECK_ES_METADATA.

**CHECK**
Performs all available checks. When you perform this action on a multistore table, it is only applied to the column store partitions of the multistore table; checks are not performed on extended storage partitions.

**CHECK_VALUE_DOMAIN**
Checks the consistency of value domains in the catalog object (such as table type, field types, and so on). When you perform this action on a multistore table, it is only applied to the column store partitions of the multistore table; checks are not performed on extended storage partitions.

**CHECK_OBJECT_REFERENTIAL_INTEGRITY**
Checks the consistency of references in the catalog object. When you perform this action on a multistore table, it is only applied to the column store partitions of the multistore table; checks are not performed on extended storage partitions.

**schema_name**
Specifies the name of the analyzed schema.

**object_name**
Specifies the name of the analyzed object.

**catalog_object_type** Specifies the type of object to analyze.

```latex
<catalog_object_type> ::= TABLE | INDEX | COLUMN | NULL
```

Use NULL to check for all extended storage objects.

Permissions

The privilege needed depends on the clause used:
• Check action – CATALOG READ or DATA ADMIN
• Repair action – DATA ADMIN

Description

The CALL CHECK_CATALOG statement performs a variety of consistency checks in both multistore and extended store tables, depending on the actions you specify as arguments to the procedure call. CHECK_CATALOG only checks the metadata of objects, and does not check the actual structure or data of tables and indexes.

Use <schema_name>, <object_name>, and <catalog_object_type> to define specific schemas or objects to check. Otherwise, enter NULL as the value for these parameters to check all objects in all schemas.

See the following:
• SAP HANA SQL and System Views Reference > SQL Reference > SQL Statements > Procedural Statements > CALL Statement (Procedural)
• SAP HANA Administration Guide > System Administration > Table and Catalog Consistency Checks > Catalog Consistency Check
• SAP Note 1977584 – Technical Consistency Checks for SAP HANA Databases

Example

You call this procedure to perform a consistency check of the tab1 table in the MYSCHEMA schema in the extended store and multistore table:

```
Sample Code
CALL CHECK_CATALOG ('CHECK_ES_METADATA', 'MYSCHEMA', 'tab1', 'table');
```

```
Output Code
SCHEMA,NAME,OBJECT_TYPE,ERROR_CODE,ERROR_MESSAGE
MYSCHEMA, TAB1, table , -301 , 'orphan table in extended storage'
```

The output is empty if there is no inconsistency.

Related Information

SAP Note 1977584
5.2.12 CALL CHECK_ES Statement (Extended Store Table)  
[Dynamic Tiering]

Checks the validity of the current database in the extended store. Optionally, it corrects allocation problems for dbspaces or databases. CHECK_ES does not check a partitioned table if partitioned data exists in offline dbspaces.

Syntax

```
call check_es ('<action>', '<object_type>', '<object_name>', '<resource_percentage>')
```

Syntax Elements

- **action**
  
The type of consistency check to perform.

```
<action> ::=  CHECK | VERIFY | ALLOCATION | ALLOCATION_LEAKED_BLOCKS | ALLOCATION_DUPLICATE_BLOCKS | ALLOCATION_UNALLOCATED_BLOCKS | DROPLEAKS
```

- **CHECK**
  
Verifies that all database pages can be read for the entire database, main cache, specific index, specific index type, specific table, or specific dbspace. If the table is partitioned, then check mode will check the table’s partition allocation bitmaps.

Run in check mode if metadata, null count, or distinct count errors are returned when running a query.

**VERIFY** Verifies the content of indexes for the entire database, or a specific index, table, or dbspace to detect keys or rows that are either missing or extras.

- **ALLOCATION**
  
Checks allocation with blockmap information for the entire database, a specific index, a specific index type, specific table, or a specific dbspace. Does not check index consistency.

- **ALLOCATION_LEAKED_BLOCKS**
  
Detects leaked blocks (allocated blocks unclaimed by any object in the specified target) for database or dbspace targets, and returns the block numbers for leaked blocks.

- **ALLOCATION_DUPLICATE_BLOCKS**
  
Detects duplicate blocks (blocks for which two or more objects claim ownership) or extra blocks (unallocated blocks owned by an object), and returns the block numbers for duplicate blocks.
**ALLOCATION_UNALLOCATED_BLOCKS**

Checks allocation for the entire database, and returns block numbers for unallocated blocks.

**DROPLEAKS**

Resets the allocation map for the entire database or dbspace.

Allocation problems can be repaired in dropleaks mode.

**object_type**

Specifies the analyzed object:

```
<object_type> ::= INDEX
             | DATABASE
             | TABLE
             | PARTITION
             | COLUMN
             | DBSPACE
```

If you specify DATABASE as your `<object_type>`, all dbspaces must be online.

**object_name**

Specifies the name of the analyzed object.

```
<object_name> ::= <indexname>
                 <table_name>
                 <column_name>
                 <dbspace_name>
                 NULL
```

**indexname**

May contain owner and table qualifiers: `[[<owner>].]<tablename>.<indexname>.

If `<owner>` is not specified, current user and database owner (`dbo`) are substituted in that order. If `<table>` is not specified, `<indexname>` must be unique.

**table_name**

May contain an owner qualifier `(<owner>.)<table_name>.

If `<owner>` is not specified, current user and database owner (`dbo`) are substituted in that order. `<tablename>` cannot be a temporary or pre-join table.

**dbspace_name**

The `<dbspace_name>` parameter contains no qualifiers. If it contains spaces, enclose it in double quotation marks.

The dbspace target examines a subset of the database’s pages that belong to that dbspace. The dbspace must be online. The dbspace and database target are semantically equivalent when the table has only one dbspace.

**NULL**
The resources percentage allows you to limit the CPU utilization of the database consistency checker by controlling the number of threads with respect to the number of CPUs. If you specify `<resource_percentage>` as 100 (the default value), then one thread is created per CPU. If you specify a value greater than 100, then there are more threads than CPUs, which might increase performance for some machine configurations. The value must be greater than 0; the minimum number of threads is 1.

Permissions

The privilege needed depends on the clause used:

- Check action – CATALOG READ or DATA ADMIN
- Repair action – DATA ADMIN

Description

CHECK_ES checks the validity of the current database in the extended store. Optionally, it corrects allocation problems for dbspaces or databases. CHECK_ES does not check a partitioned table if partitioned data exists in offline dbspaces.

Enclose the CHECK_ES parameter string in single quotes. It cannot be greater than 255 bytes in length.

If an identifier contains a dot, enclose object names in square brackets. For example:

```
call CHECK_ES('allocation','table','[SCH_A].[EXT.A]','100')
```

Depending on the execution mode, CHECK_ES output includes summary results, errors, informational statistics, and repair statistics. Error statistics are indicated by asterisks (****), and appear only if errors are detected.

If a CHECK_ES call returns an error, use the ES_REBUILD_INDEX procedure to rebuild indexes for tables in extended storage, including extended store partitions of a multistore table.

For more information, see SAP HANA SQL and System Views Reference > SQL Reference > SQL Statements > Procedural Statements > CALL Statement (Procedural)

Examples

This procedure checks the consistency of block allocation for the entire database, allocating a resource percentage of 100 percent (which creates one thread per CPU):

```
'Sample Code

Call CHECK_ES ( 'allocation', 'database', '', '100' );
```
5.2.13 CALL CHECK_TABLE_CONSISTENCY Statement (Multistore Table) [Dynamic Tiering]

Performs consistency check actions in the column store partition of a multistore table.

Syntax

```
CALL CHECK_TABLE_CONSISTENCY ('<action>', '<schema name>', '<table name>')
```
Syntax Elements

[action]
The type of consistency check to perform.

<action> ::=   CHECK
| CHECK_COLUMN_TABLES
| CHECK_PRIMARY_KEY
| CHECK_ROWID
| CHECK_PARTITIONING
| CHECK_PARTITIONING_DATA
| REPAIR_PARTITIONING_DATA

CHECK
Performs all available checks.
CHECK_COLUMN_TABLES
Performs a check restricted to default (column) store and multistore tables.
CHECK_PRIMARY_KEY
Checks the consistency of the primary key in the default store.
CHECK_ROWID
Checks the consistency of the internal $rowid$ column only for the column store partition of a multistore table.
CHECK_PARTITIONING
Checks the consistency of partitioning-related metadata.
CHECK_PARTITIONING_DATA
Checks the assignment of rows to partitions.
REPAIR_PARTITIONING_DATA
Repairs the assignment to rows to partitions.

Additional <action> options are available for the default store.

/schema_name
Specifies the name of the analyzed schema.

/object_name
Specifies the name of the analyzed object.
/table_name
Specifies the name of the analyzed table.

Permissions

The privilege needed depends on the clause used:

- Check action – CATALOG READ or DATA ADMIN
• Repair action – DATA ADMIN

Description

CHECK_TABLE_CONSISTENCY performs consistency check actions in the column store partition of a multistore table.

You can only perform a table consistency check for column store partitions in a multistore table.

If you have tables or schemas with names containing lowercase characters, slashes ("/"), or other special characters, enclose them in double-quotation marks to prevent CHECK_TABLE_CONSISTENCY from normalizing them, treating them as strings.

For more information about CALL and table consistency checks, see:

• SAP HANA SQL and System Views Reference > SQL Reference > SQL Statements > Procedural Statements > CALL Statement (Procedural)
• SAP HANA Administration Guide > System Administration > Table and Catalog Consistency Checks > Table Consistency Checks
• SAP Note 1977584 – Technical Consistency Checks for SAP HANA Databases

Example

This procedure checks the consistency for a table named "ABC/abc" in the SYSTEM schema:

CALL CHECK_TABLE_CONSISTENCY ('CHECK', 'SYSTEM', '"ABC/abc"');

Related Information

SAP Note 1977584
CALL Statement (Procedural) [page 715]
5.2.14 CALL ES_REBUILD_INDEX Statement (Multistore Tables) [Dynamic Tiering]

When you encounter reports or errors running the CHECK_ES extended store database consistency checker, or if an index fails to meet performance expectations, call this procedure to rebuild indexes for tables in extended storage, including extended store partitions of a multistore table.

Syntax

```
CALL ES_REBUILD_INDEX ('<SCHEMA_NAME>', '<TABLE_NAME>', '<OBJECT_NAME>', '<OBJECT_TYPE>', '<ESTIMATED_CARDINALITY>')
```

Syntax Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>The schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>The table name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>The object name.</td>
</tr>
<tr>
<td>OBJECT_NAME ::=</td>
<td>`{ COL_NAME</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>The object type.</td>
</tr>
<tr>
<td>OBJECT_TYPE ::=</td>
<td>`{ INDEX</td>
</tr>
<tr>
<td>INDEX – for building HG indexes</td>
<td></td>
</tr>
<tr>
<td>COLUMN – for building FP indexes</td>
<td></td>
</tr>
<tr>
<td>PRIMARY_KEY – for rebuilding system-created primary key indexes</td>
<td></td>
</tr>
<tr>
<td>ESTIMATED_CARDINALITY</td>
<td>The estimated cardinality.</td>
</tr>
<tr>
<td>ESTIMATED_CARDINALITY ::=</td>
<td>`{ INTEGER</td>
</tr>
</tbody>
</table>

Use NULL when `<OBJECT_NAME>` is PRIMARY_KEY.

Use INTEGER when OBJECT_TYPE is COLUMN, and you are creating an Nbit index with non-default distinct count. Otherwise, use NULL.
Permissions

You must have the DATA ADMIN privilege.

Examples

- The following example rebuilds column-indexes:
  
  
  ```sql
  CALL ES_REBUILD_INDEX ( 'SCH_1', 'TAB_1', 'COL_1', 'COLUMN', NULL )
  ```

- The following example rebuilds column indexes with distinct count:
  
  ```sql
  CALL ES_REBUILD_INDEX ( 'SCH_1', 'TAB_1', 'COL_1', 'COLUMN', '1024' )
  ```

- The following example rebuilds user-created HG indexes:
  
  ```sql
  CALL ES_REBUILD_INDEX ( 'SCH_1', 'TAB_1', 'IDX_1', 'INDEX', NULL )
  ```

- The following example rebuilds primary key-related HG indexes:
  
  ```sql
  CALL ES_REBUILD_INDEX ( 'SCH_1', 'TAB_1', NULL, 'PRIMARY_KEY', NULL )
  ```

5.2.15 CREATE AUDIT POLICY Statement [Dynamic Tiering]

Creates a new audit policy for extended tables.

Syntax

```
CREATE AUDIT POLICY <policy_name> AUDITING <audit_status>
<audit_actions> LEVEL <audit_level> <opt_audit_trail_type>
```​

Syntax Elements

- **policy_name**
  
  Specifies the name of the audit policy to be created.
  
  ```sql
  <policy_name> ::= <identifier>
  ```

- **audit_status**
  
  Defines whether successful, unsuccessful or all executions of the specified audit actions are audited.
  
  ```sql
  <audit_status> ::= 
  ```
SUCCESSFUL
| UNSUCCESSFUL
| ALL

**audit_actions**

Specifies the audit actions for the audit policy.

```xml
<audit_actions> ::=  
  <actions_for_user>  
  <audit_action_list>  
  <target_audit_action_list>
```

**actions_for_user**

Specifies that commands executed by a user or a set of users or all users except the named set of users are audited.

```xml
<actions_for_user> ::= ACTIONS [EXCEPT] FOR <user_name>[, <user_name>...] 
```

**user_name**

Specifies the username of the user to be audited by the audit policy.

```xml
[user_name] ::= <identifier>
```

**audit_action_list**

Audits specific system actions, optionally limited to a user or a set of users or audited for all users except for the given set of users.

```xml
<audit_action_list> ::=   
  <audit_action_name>[, <audit_action_name>]...   
  [ [EXCEPT] FOR <user_name>[, <user_name>...] ] 

<audit_action_name> ::=  CREATE EXTENDED STORAGE  
| ALTER EXTENDED STORAGE  
| DROP EXTENDED STORAGE
```

For more information on audit actions, see the table of audit actions provided in the **Description** section of this topic.

**target_audit_action_list**

Audits actions on a database object or set of objects. Optionally this auditing can be limited to a user or a set of users or audited for all users except for the given set of users.

```xml
<target_audit_action_list> ::=  
  <target_audit_action_entry> [ [EXCEPT] FOR <user_name>[, <user_name>...] ] 

<target_audit_action_entry> ::=  
  <audit_action_name>[, <audit_action_name>]...  
  ON <audit_object_name>[, <audit_object_name>]... 

<audit_action_name> ::=  INSERT  
| UPDATE  
| DELETE  
| SELECT  
| EXECUTE
```

Only objects of type table, view, procedure, and schema can be specified in the `<target_audit_action_entry>`. In case of `<schema_name>.*`, all objects that are stored in this schema and can be combined with the specified `<target_audit_action_name>` values are...
audited. Synonyms and sequences cannot be selected as objects for audit policies. Only specified `<target_audit_action_name>` values can be combined with an object.

**audit_object_name**

```plaintext
<audit_object_name> ::=  
<table_name>  
| <view_name>  
| <procedure_name>  
| <schema_name>.*  
<table_name> ::= [<schema_name>.]<identifier>  
<procedure_name> ::= [<schema_name>.]<identifier>  
<schema_name> ::= <unicode_name>
```

Specifies a database object for the target audit action.

**audit_level**

Assigns an audit policy to an audit level.

```plaintext
<audit_level> ::=  
EMERGENCY  
| ALERT  
| CRITICAL  
| WARNING  
| INFO
```

Possible levels, in decreasing order of importance, are:

- EMERGENCY
- ALERT
- CRITICAL
- WARNING
- INFO

**opt_audit_trail_type**

Specifies the audit trail target type(s) for the audit policy.

```plaintext
<opt_audit_trail_type> ::= TRAIL TYPE <audit_trail_type_list>  
<audit_trail_type_list> ::= <audit_trail_type_name> [,  
<audit_trail_type_name> [,... ] ]  
<audit_trail_type_name> ::=  
TABLE  
| SYSLOG  
| CSV
```

**Permissions**

You must have the AUDIT ADMIN privilege.
Description

The CREATE AUDIT POLICY statement for dynamic tiering creates a new audit policy. This audit policy can then be enabled and results in the auditing of the specified audit actions to occur.

The specified audit policy name must be unique and not match the name of an existing audit policy.

An audit policy defines which audit actions are audited. Audit policies need to be enabled for auditing to occur.

One audit policy can contain one of the following:

- Non-restricted auditing for n (>=1) users
- Auditing for actions not restricted to objects
- Auditing for actions which are restricted to objects.

For the last two alternatives listed, an optional restriction for user(s) is available.

Audit actions (like CREATE/ALTER/DROP/START/STOP DATABASE) can only be specified in the systemDB.

For auditing to occur, audit policies have to be created and enabled. Also the configuration parameter `global_auditing_state` (see Configuration Parameters) has to be set to true.

One or more audit trail targets can be specified for an audit policy at the time of creation or after creation (see ALTER AUDIT POLICY [Dynamic Tiering] [page 1292]).

The allowed audit trail targets are:

- SYSLOG: uses the system syslog.
- TABLE: stores audit information in database table. The audit log is accessible using AUDIT_LOG system view.
- CSV: stores audit information as comma-separated values in a text file. Should be used only for testing purposes.

Audit Actions

The table below contains the available audit actions. Each of the audit actions is in a specific group, audit actions in the same group can be combined into one audit policy. At any time, you can retrieve the list of possible audit actions mentioned in this table by executing the following statement: `SELECT * FROM AUDIT_ACTIONS ORDER BY action_group, action_name;`.

<table>
<thead>
<tr>
<th>Audit Action Name</th>
<th>Group Number</th>
<th>Audit Operation</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE EXTENDED STORAGE</td>
<td>20</td>
<td>creation of the extended store</td>
<td></td>
</tr>
<tr>
<td>ALTER EXTENDED STORAGE</td>
<td>20</td>
<td>changes to the extended store</td>
<td></td>
</tr>
<tr>
<td>DROP EXTENDED STORAGE</td>
<td>20</td>
<td>deletion of the extended store</td>
<td></td>
</tr>
</tbody>
</table>
Configuration Parameters

Currently the configuration parameter for auditing is stored in global.ini configuration file, in the auditing configuration section.

```
global_auditing_state ( 'true' / 'false' )
```

Activates / deactivates auditing globally, regardless of the enabled state of the audit policies. The default is false, meaning: no auditing occurs.

```
default_audit_trail_type ( 'SYSLOGPROTOCOL' / 'CSTABLE' / 'CSVTEXTFILE' )
```

Specifies how to store the auditing results. SYSLOGPROTOCOL is the default.

- SYSLOGPROTOCOL: uses the system syslog.
- CSTABLE: stores audit information in database table. The audit log is accessible using AUDIT_LOG system view.
- CSVTEXTFILE: stores audit information as comma-separated values in a text file. Should be used only for testing purposes.

```
default_audit_trail_path
```

Specifies the audit file storage location for the CSVTEXTFILE audit trail type.

```
emergency_audit_trail_type
```

Specifies the audit trail target(s) for the events with severity EMERGENCY.

```
alert_audit_trail_type
```

Specifies the audit trail target(s) for the events with severity ALERT.

```
critical_audit_trail_type
```

Specifies the audit trail target(s) for the events with severity CRITICAL.

Multiple audit trail targets could be specified for parameters `emergency_audit_trail_type` / `alert_audit_trail_type` / `critical_audit_trail_type` by separating the trail type names with comma. To store the auditing information for events with severity EMERGENCY to both Syslog and database table, set the value of `emergency_audit_trail_type` to 'CSTABLE,SYSLOGPROTOCOL'.

As for all configuration parameters, these parameters can be selected in view `M_INIFILE_CONTENTS`, assuming that the current user has the required privileges. These parameters can only be seen if they have been explicitly set.

Example

You create a new audit policy named `ext_store_admin_policy` to audit successful attempts to create, alter, or drop extended storage. The audit policy has the medium audit level CRITICAL.
This policy has to be explicitly enabled (see ALTER AUDIT POLICY [Dynamic Tiering] [page 1292]) to cause the auditing of the audit policy.

```
CREATE AUDIT POLICY ext_store_admin_policy CREATE EXTENDED STORAGE, ALTER EXTENDED STORAGE, DROP EXTENDED STORAGE LEVEL CRITICAL
```

You enable auditing of INSERT and UPDATE on an extended table:

```
CREATE SCHEMA MY_SCHEMA OWNED BY SYSTEM;
CREATE COLUMN TABLE MY_SCHEMA.MY_EXT_TABLE (A INT PRIMARY KEY, B INT)
USING EXTENDED STORAGE;
CREATE AUDIT POLICY EXT_OBJECT_AUDIT
AUDITING SUCCESSFUL INSERT, UPDATE ON MY_SCHEMA.MY_EXT_TABLE
LEVEL INFO;
ALTER AUDIT POLICY EXT_OBJECT_AUDIT ENABLE;
INSERT into MY_SCHEMA.MY_EXT_TABLE values ( 1, 2 );
INSERT into MY_SCHEMA.MY_EXT_TABLE values ( 3, 4 );
UPDATE MY_SCHEMA.MY_EXT_TABLE SET A = 2 WHERE A = 3;
```

Related Information

- AUDIT_LOG System View [page 1501]
- AUDIT_POLICIES System View [page 1503]
- M_INIFILE_CONTENTS System View [page 1942]

5.2.16 CREATE EXTENDED STORAGE Statement [Dynamic Tiering]

Creates a new extended storage database, starts it, and registers it as an SAP HANA service.

Syntax

```
CREATE EXTENDED STORAGE [ AT [LOCATION] '<host-string>' ] SIZE <size-value> (KB|MB|GB|TB) [(ENABLE|DISABLE) DELTA]
```

Syntax Elements

- **host-string**
  Specifies the host on which extended storage is being created. The host name must exactly match the host name that appears in the M_SERVICES view.

  `<host-string> ::= <string_literal>\n`
size-value

Specifies the size of data to be managed in extended storage. Initial size cannot exceed 1 TB. The size of any individual dbfile added to a dynamic tiering database cannot exceed 1 TB.

```
<size-value> ::= <int_literal>
```

ENABLE | DISABLE DELTA

Specifies whether RLV store is created on extended storage. ENABLE DELTA supports row-level versioning. The default is DISABLE DELTA (table-level versioning).

```
ENABLE | DISABLE DELTA
```

Permissions

You must have the EXTENDED STORAGE ADMIN privilege.

Description

The CREATE EXTENDED STORAGE statement creates a new extended storage database, starts it, and registers it as an SAP HANA service.

i Note

If you create extended storage, you cannot then enable data volume encryption in the SAP HANA database. You can, however, first enable data volume encryption in the SAP HANA database before creating extended storage. Alternatively, you can convert an existing unencrypted SAP HANA database with extended storage by performing a backup and restore. See Extended Storage Database Encryption in the SAP HANA Dynamic Tiering: Administration Guide for more information.

Caution

Creating extended storage after a full data backup, but before recovery, will break business continuity. See 2375865.

Example

This creates an extended storage of size 10 GB on host dbhost.example.com with delta capability enabled.

```
CREATE EXTENDED STORAGE AT 'dbhost.example.com' size 10 GB ENABLE DELTA
```
5.2.17 CREATE INDEX Statement (Extended Store Table)  
[Dynamic Tiering]

Creates an index on an extended store table column.

Syntax

```
CREATE [UNIQUE] [BTREE | CPBTREE] INDEX <index_name> ON <table_name>  
({<column_name_order_entry>|[<column_name_order_entry>]}...)  
[<global_index_order>] [fillfactor]  
[NOWAIT]
```

Syntax Elements

**UNIQUE** Defines that a unique index should be created. A duplicates check occurs when the index is created and when a record is added to the table.

**BTREE** Defines that a unique index should be created. A duplicates check occurs when the index is created and when a record is added to the table.

**CPBTREE** Specifies a CPB+ tree index type. CPB+ tree stands for Compressed Prefix B+ tree; this index tree type is based on pkB tree. CPB+ tree is a very small index because it uses partial key that is only part of full key in index nodes. CPB+ tree shows better performance than B+ tree for larger keys.

**index_name**

Specifies the name of the index to be created, with optional schema name.

```
<index_name> ::= [<schema_name>.]<identifier>  
<schema_name> ::= <identifier>
```

**column_name_order_entry** Specifies the columns to be used in the index, with an optional ordering.

```
<column_name_order_entry> ::= <column_name> [column_order]
```

**column_order** Specifies whether the column index should be created in ascending or descending order. The default ordering is ASC.

```
<column_order> ::= ASC | DESC
```

**global_index_order** Specifies the index ordering for all columns in the index.

```
<global_index_order> ::= ASC | DESC
```

**i Note**

<column_order> and <global_index_order> cannot be used when <global_index_order> is used.
**fillfactor** Specifies how each node of a new index is filled. It is a percentage value of integer from 50 to 100, and the default value is 90.

\[
<\text{fillfactor}> ::= \text{FILLFACTOR} \ <\text{unsigned_integer}>
\]

**NOWAIT**

Specifies that CREATE INDEX statement returns an error immediately if a table lock cannot be acquired.

\[
<\text{NOWAIT}>
\]

**Permissions**

You must have the INDEX object privilege.

**Description**

The CREATE INDEX statement creates an index on an extended store table column.

Only BTREE and CPBREE index types are supported on extended store tables.

**Example**

You create Table `t`, and then add index `idx` on column `b` of table `t` with ascending order.

```
CREATE ROW TABLE t (a INT, b NVARCHAR(10), c NVARCHAR(20));
CREATE INDEX idx ON t(b);
```

Create a CPBTREE index `idx1` on column `a` of table `t` with ascending order and column `b` with descending order.

```
CREATE CPBTREE INDEX idx1 ON t(a, b DESC);
```

Create a CPBTREE index `idx2` on column `a` and `c` of table `t` with descending order.

```
CREATE INDEX idx2 ON t(a, c) DESC;
```

Create a UNIQUE CPBTREE index `idx3` on column `b` and `c` of table `t` with ascending order.

```
CREATE UNIQUE INDEX idx3 ON t(b, c);
```

Create a UNIQUE BTREE index `idx4` on column `a` of table `t` with ascending order.

```
CREATE UNIQUE INDEX idx4 ON t(a);
```
5.2.18 CREATE INDEX Statement (Multistore Table)
[DYNAMIC TIERING]

Creates an index on a multistore table column.

Syntax

```
CREATE [UNIQUE] [BTREE | CPBTREE] INDEX <index_name> ON <table_name>     
{<column_name_order_entry>[(, <column_name_order_entry>)]...}  
[<global_index_order>] [<fillfactor>]  
[NOWAIT] [FOR {DEFAULT|EXTENDED} STORAGE]
```

Syntax Elements

**UNIQUE** Defines that a unique index should be created. A duplicates check occurs when the index is created and when a record is added to the table.

**BTREE** Specifies a B+ tree index type. B+ tree uses an index tree that maintains sorted data in a way that performs efficient insertion, deletion, and search of records.

**CPBTREE** Specifies a CPB+ tree index type. CPB+ tree stands for Compressed Prefix B+ tree; this index tree type is based on PKB tree. CPB+ tree is a very small index because it uses **partial key** that is only part of full key in index nodes. CPB+ tree shows better performance than B+ tree for larger keys.

**index_name**

Specifies the name of the index to be created, with optional schema name.

```
<index_name> ::= [<schema_name>.]<identifier>  
<schema_name> ::= <identifier>
```

**column_name_order_entry** Specifies the columns to be used in the index, with an optional ordering.

```
<column_name_order_entry> ::= <column_name> [<column_order>]
```

**column_order** Specifies whether the column index should be created in ascending or descending order. The default ordering is ASC.

```
<column_order> ::= ASC | DESC
```
**global_index_order** Specifies the index ordering for all columns in the index.

```sql
<global_index_order> ::= ASC | DESC
```

**Note**

`<column_order>` and `<global_index_order>` cannot be used when `<global_index_order>` is used.

**fillfactor** Specifies how each node of a new index is filled. It is a percentage value of integer from 50 to 100, and the default value is 90.

```sql
<fillfactor> ::= FILLFACTOR <unsigned_integer>
```

**NOWAIT**

Specifies that `CREATE INDEX` statement returns an error immediately if a table lock cannot be acquired.

```sql
<NOWAIT>
```

**FOR {DEFAULT | EXTENDED} STORAGE**

Specifies that `CREATE INDEX` creates a partial index on one store or the other. `FOR DEFAULT STORAGE` creates a partial index on the default store. `FOR EXTENDED STORAGE` creates a partial index on the extended store. If you don't specify the `FOR STORAGE` clause, the statement creates a regular index on default and extended stores instead of a partial index.

```sql
<FOR {DEFAULT | EXTENDED} STORAGE >
```

**Permissions**

You must have the `INDEX` object privilege.

**Description**

The `CREATE INDEX` statement creates an index on a multistore table.

Only BTREE and CPBREE index types are supported on multistore tables.

**Example**

Create multistore table `ext_a`:

```sql
CREATE COLUMN TABLE ext_a (c1 INT, c2 INT)
PARTITION BY RANGE (c1) (USING DEFAULT STORAGE
(PARTITION 0 <= values < 10000, PARTITION 10000 <= values < 1000000)
USING EXTENDED STORAGE (PARTITION 1000000 <= values < 100000000));
```
Create a partial index on the default store only:

```sql
CREATE BTREE INDEX idx_a ON ext_a (c1) FOR DEFAULT STORAGE;
```

Drop the partial index from the default store and create an index again on extended storage:

```sql
DROP INDEX idx_a;
CREATE BTREE INDEX idx_a ON ext_a (c1) FOR EXTENDED STORAGE;
```

**Related Information**

INDEXES System View [page 1585]
INDEX_COLUMNS System View [page 1587]

### 5.2.19 CREATE STATISTICS Statement (Extended Store Table) [Dynamic Tiering]

Creates data statistic objects that allow the query optimizer to make better decisions for query plans.

**Syntax**

```sql
CREATE STATISTICS [<data_statistics_name>] ON <data_sources>
[ [<data_statistics_type> ] ]
[ <data_statistics_properties> ]
[ <initial_refresh> ];
```

**Syntax Element**

- **data_statistics_name**
  Specifies a unique name for the data statistics object.

  ```sql
  <data_statistics_name> ::= [ <schema_name>.]<identifier>
  ```

  `<data_statistics_name>` is only allowed when the result of the creation is a single data statistics object. The number of data statistics objects created by CREATE STATISTICS is determined by the combination of `<data_statistics_type>` and the number of columns specified in `<data_sources>`.

  ```sql
  <schema_name> ::= <identifier>
  ```

  `<schema_name>` must be the same as specified for data source.

  **data_sources**
Specifies the data sources you want to create data statistics objects for.

\[
<\text{data\_sources}> ::= \text{table\_name} \[ ( \text{column\_name}[, \text{column\_name}][,...] ) \]
\]

For RECORD COUNT data statistics objects, you cannot specify columns as part of data_sources.

\[
\text{table\_name}
\]

Specifies the table name you want to create statistics on.

\[
<\text{table\_name}> ::= [<\text{schema\_name}>.]<\text{identifier}>
\]

If you specify a data statistics name, then you can only specify one data source.

\[
\text{column\_name}
\]

Specifies the column for which the data statistics are defined.

\[
<\text{column\_name}> ::= <\text{identifier}>
\]

If no <column_name> is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

Specifying multiple columns when creating single-column data statistics is equivalent to executing a matching CREATE STATISTICS statement for each column individually. If you specify a data statistics name, then you can only specify one data source.

\[
\text{data\_statistics\_type}
\]

Specifies the type of data statistics object to create.

\[
<\text{data\_statistics\_type}> ::= \text{TYPE }<\text{type\_name}>
<\text{type\_name}> ::= \text{HISTOGRAM} | \text{SIMPLE} | \text{TOPK} | \text{SKETCH} | \text{SAMPLE} | \text{RECORD COUNT}
\]

A data source can have only one data statistics object of a certain type. For example, column A of table T can have one data statistics object of type HISTOGRAM and one of type SIMPLE. If the TYPE clause is not specified, then the default is HISTOGRAM. Some data statistic types may not be appropriate for a given data source.

**HISTOGRAM**

Creates a data statistics object that helps the query optimizer estimate the data distribution in a single-column data source. If you specify multiple columns in <data_sources>, then multiple data statistics objects (HISTOGRAM) are created--one per column specified.

**SIMPLE**

Creates a data statistics object that helps the query optimizer calculate basic statistics, such as min, max, null count, count, and distinct count for a single-column data source. If you specify multiple columns in <data_sources>, then multiple data statistics objects are created--one per column specified.

**TOPK**
Creates a data statistics object that helps the query optimizer identify the highest-frequency values in a table data source. If you specify multiple columns in `<data_sources>`, then multiple data statistics objects are created—one per column specified.

**SKETCH**

Creates a data statistics object that helps the query optimizer estimate the number of distinct values in the data source. A data statistics object is created for the specified `<table_name>(<column-name>, . . . )`, which approximates the number of distinct tuples in the projection of the table on the set of specified columns.

**SAMPLE**

Creates a sample of data from `<data_source>` that the SQL optimizer can use during optimization. When beneficial, the SQL optimizer generates system SAMPLE data statistics objects automatically on column and row store tables. However, this behavior can incur a cost to performance. You can avoid this cost by creating SAMPLE data statistics objects explicitly (in advance). Creating them explicitly is especially useful in situations where sampling live table data is expensive (for example, very large tables).

**RECORD COUNT**

Creates a data statistics object that helps the query optimizer calculate the number of records (rows) in a table data source. The RECORD COUNT type is a table-wide statistic. You do not specify columns in `<data_sources>` when creating a RECORD COUNT data statistics object. When beneficial, the SQL optimizer maintains system RECORD COUNT data statistics objects automatically on column and row store tables.

**data_statistics_properties**

Specifies the properties of the data statistics objects.

```
<data_statistics_properties> ::= 
    <data_statistics_property>[,<data_statistics_property>[,...]].
<data_statistics_property> ::= 
    REFRESH TYPE <refresh_type>
    ENABLE <on_off>
    BUCKETS <unsigned_integer>
    {MEMORY <memory_bytes> | MEMORY PERCENT <memory_percentage> }
    CONSTRAINT <constraint_param>
```

Restrictions to which properties apply to which statistic types are noted in the property descriptions.

**REFRESH TYPE refresh_type**

Specifies the strategy for the data statistics object.

```
<refresh_type> ::= ( AUTO | MANUAL | DEFAULT )
```

AUTO specifies that the data statistics object is refreshed automatically when underlying data changes. AUTO is only supported on column store, extended store, and multistore tables.

MANUAL specifies that the database statistics object is not refreshed until a rebuild is explicitly requested by a REFRESH STATISTICS statement.

DEFAULT specifies that the database server decides the best refresh strategy based on the data source. For example, for data statistics objects on column store data sources, the database server applies AUTO for the default.

REFRESH TYPE only affects data statistics objects that are enabled.
ENABLE on_off

Controls whether the optimizer uses the data statistics object.

\[
\text{on_off} ::= \{ \text{ON} \mid \text{OFF} \}
\]

ENABLE ON enables the optimizer to see the data statistics object. The data statistics object must be populated with data for the optimizer to use it. ENABLE ON specified with NO INITIAL REFRESH returns an error.

ENABLE ON is the default behavior.

ENABLE OFF disables the use of the data statistics object by the optimizer and prevents the ability to refresh the data statistics object. Data statistics objects that are not enabled can still be dropped. To make a data statistics object with ENABLE OFF accessible to the optimizer, execute an ALTER STATISTICS...ENABLE ON statement.

BUCKETS unsigned_integer

The BUCKETS property is only for use with TYPE HISTOGRAM or TOPK. For HISTOGRAM, BUCKETS specifies the maximum number of data buckets in the HISTOGRAM. For TOPK, BUCKETS specifies the K value.

The default is automatically determined by the data statistics building algorithm in use.

If a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

For column store, extended store, and multistore tables only, if a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

MEMORY memory_bytes

Specifies the maximum amount of memory, in bytes, to use for QOPTIMAL HISTOGRAMS.

\[
\text{memory_bytes} ::= \text{unsigned_integer}
\]

The MEMORY parameter limits the memory for QOPTIMAL HISTOGRAMS. MEMORY applies only to the QOPTIMAL HISTOGRAM algorithm. Small values for MEMORY may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

MEMORY PERCENT memory_percentage

Specifies the maximum amount of memory to use for the data statistics object, expressed as a percentage of the space used by the data source.

\[
\text{memory_percentage} ::= \text{unsigned_integer}
\]

HISTOGRAMS can use a large amount of memory for some data sources. \(<\text{memory_percentage}>\) represents the maximum amount of memory that can be used for the data statistics object. For example, if a data source is a table column that uses 100 MB of memory, and \(<\text{memory_percentage}>\) is 5, then the data statistics object for this column can use, at most, 5 MB for its in-memory representation.
The default is automatically determined by the HISTOGRAM algorithm in use. Small values for MEMORY PERCENT may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

**CONSTRAINT**
Specifies constraints to use for the specified for type HISTOGRAM.

```plaintext
CONSTRAINT <constraint_param>  
<constraint_param> ::= MAXDIFF
```

Histogram sizing restrictions (BUCKETS, MEMORY, and MEMORY PERCENT) are applied per HISTOGRAM.

A non-default CONSTRAINT for a HISTOGRAM returns an error.

**initial_refresh**
Specifies whether to populate the data statistics object with data after creation.

```plaintext
<initial_refresh> ::= [ NO ] INITIAL REFRESH
```

**INITIAL REFRESH**
Creates the definition of the data statistics object and populates it with data. The default behavior is INITIAL REFRESH.

**NO INITIAL REFRESH**
Creates the definition of the data statistics object, but does not populate it with data.

Use NO INITIAL REFRESH when you want to change the underlying data before refreshing the data statistics object.
You cannot specify NO INITIAL REFRESH if ENABLE OFF is not specified.

**Permissions**

One of the following is true with regards to required permission:

- You own the table you are creating statistics on.
- You have the ALTER privilege on the object you are creating statistics on.

**Description**

CREATE STATISTICS statement creates data statistics objects that approximate the specified `<data_sources>`. Data statistics objects are used to estimate the properties of the data without directly accessing the data itself. This can be useful during query optimization, query execution, and so on.
Examples

You create a MAXDIFF HISTOGRAM with a maximum of 1000 buckets on the column T(X), where T is an extended storage table:

```
CREATE STATISTICS ON MYSYSTEM.T(X) TYPE HISTOGRAM BUCKETS 1000
```

You create a MAXDIFF HISTOGRAM on the column T(X), where T is an extended store table. The BUCKETS value that is not specified is determined by the HISTOGRAM build algorithm. You specify the name of this data as "HISTOGRAM_T_X":

```
CREATE STATISTICS "HISTOGRAM_T_X" ON MYSYSTEM.T(X) TYPE HISTOGRAM
```

Create a SKETCH on T(A, B):

```
CREATE STATISTICS ON MYSYSTEM.T(A,B) TYPE SKETCH;
```

Related Information

ALTER STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1299]
DROP STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1393]
REFRESH STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1437]
M_DATA_STATISTICS System View [page 1851]
M_SYSTEM_DATA_STATISTICS System View [page 2189]

5.2.20 CREATE STATISTICS Statement (Multistore Table) [Dynamic Tiering]

Creates data statistic objects that allow the query optimizer to make better decisions for query plans.

Syntax

```
CREATE STATISTICS [<data_statistics_name>] ON <data_sources>
  [ <data_statistics_type> ]
  [ <data_statistics_properties> ]
  [ <initial_refresh> ];
```

Syntax Element

```
data_statistics_name
```
Specifies a unique name for the data statistics object.

\[
<\text{data\_statistics\_name}> ::= [ <\text{schema\_name}>.]<\text{identifier}>
\]

\(<\text{data\_statistics\_name}>\) is only allowed when the result of the creation is a single data statistics object. The number of data statistics objects created by CREATE STATISTICS is determined by the combination of \(<\text{data\_statistics\_type}>\) and the number of columns specified in \(<\text{data\_sources}>\).

\[
<\text{schema\_name}> ::= <\text{identifier}>
\]

\(<\text{schema\_name}>\) must be the same as specified for data source.

\section*{data\_sources}

Specifies the data sources you want to create data statistics objects for.

\[
<\text{data\_sources}> ::= \\
<\text{table\_name}> \{ ( <\text{column\_name}>, <\text{column\_name}>, ... ) \} \} \quad \text{FOR \{DEFAULT | EXTENDED\} STORAGE}
\]

For RECORD COUNT data statistics objects, you cannot specify columns as part of \(<\text{data\_sources}>\).

\section*{table\_name}

Specifies the table name you want to create statistics on.

\[
<\text{table\_name}> ::= [<\text{schema\_name}>.]<\text{identifier}>
\]

If you specify a data statistics name, then you can only specify one data source.

\section*{column\_name}

Specifies the column for which the data statistics are defined.

\[
<\text{column\_name}> ::= <\text{identifier}>
\]

If no \(<\text{column\_name}>\) is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

Specifying multiple columns when creating single-column data statistics is equivalent to executing a matching CREATE STATISTICS statement for each column individually. If you specify a data statistics name, then you can only specify one data source.

\section*{FOR \{DEFAULT | EXTENDED\} STORAGE}

Specify DEFAULT when the data statistics object is created on the column store part. Specify EXTENDED STORAGE when the data statistics object is created on the extended storage part. If you do not specify anything, then two data statistics objects are created, one on each part. A table column to use as the data source.

Statistics of some types can only be created on one storage class partition; for example, SKETCH on HANA column store partitions, and RECORD COUNT on extended storage partitions. Attempting to create such objects on a multistore table without specifying storage type returns an error.

You can implement different statistics types for different storage classes, and create statistics to address specific performance problems that only affect the query plan for one of the storage classes of the table. The statement returns an error if no partitions of the specified storage type exist.

\section*{data\_statistics\_type}
Specifies the type of data statistics object to create.

```
<data_statistics_type> := TYPE <type_name>
?type_name> ::=  
  HISTOGRAM
  | SIMPLE
  | TOPK
  | SKETCH
  | SAMPLE
  | RECORD COUNT
```

A data source can have only one data statistics object of a certain type. For example, column A of table T can have one data statistics object of type HISTOGRAM and one of type SIMPLE. If the TYPE clause is not specified, then the default is HISTOGRAM. Some data statistic types may not be appropriate for a given data source.

**HISTOGRAM**

Creates a data statistics object that helps the query optimizer estimate the data distribution in a single-column data source. If you specify multiple columns in `<data_sources>`, then multiple data statistics objects (HISTOGRAM) are created—one per column specified.

**SIMPLE**

Creates a data statistics object that helps the query optimizer calculate basic statistics, such as min, max, null count, count, and distinct count for a single-column data source. If you specify multiple columns in `<data_sources>`, then multiple data statistics objects are created—one per column specified. SIMPLE statistics are only supported on the extended storage part of a multistore table and must be created using the FOR EXTENDED STORAGE clause.

**TOPK**

Creates a data statistics object that helps the query optimizer identify the highest-frequency values in a table data source. If you specify multiple columns in `<data_sources>`, then multiple data statistics objects are created—one per column specified. TOPK statistics are only supported on the extended storage part of a multistore table and must be created using the FOR EXTENDED STORAGE clause.

**SKETCH**

Creates a data statistics object that helps the query optimizer estimate the number of distinct values in the data source. A data statistics object is created for the specified `<table_name>(<column-name>,...)`, which approximates the number of distinct tuples in the projection of the table on the set of specified columns.

**SAMPLE**

Creates a sample of data from `<data_source>` that the SQL optimizer can use during optimization. When beneficial, the SQL optimizer generates system SAMPLE data statistics objects automatically on column and row store tables. However, this behavior can incur a cost to performance. You can avoid this cost by creating SAMPLE data statistics objects explicitly (in advance). Creating them explicitly is especially useful in situations where sampling live table data is expensive (for example, very large tables).

**RECORD COUNT**

Creates a data statistics object that helps the query optimizer calculate the number of records (rows) in a table data source. The RECORD COUNT type is a table-wide statistic. You do not specify columns in `<data_sources>` when creating a RECORD COUNT data statistics object. When beneficial, the SQL optimizer maintains system RECORD COUNT data statistics objects automatically on column and row store tables. RECORD COUNT statistics can only be created on the extended storage part of a table.
multistore table using the FOR EXTENDED STORAGE clause. They cannot be created on the default storage portion of the table.

**data_statistics_properties**

Specifies the properties of the data statistics object.

```
<data_statistics_properties> ::=    <data_statistics_properties>,<data_statistics_property>[,...] 

<data_statistics_property> ::=    REFRESH TYPE <refresh_type>
                               | ENABLE <on_off>
                               | BUCKETS <unsigned_integer>
                               | QERROR <numeric_literal>
                               | QTHETA <unsigned_integer>
                               | { MEMORY <memory_bytes> | MEMORY PERCENT <memory_percentage> }
                               | ACCURACY <numeric_literal>
                               | PREFIX BITS <unsigned_integer>
                               | PERSISTENT <on_off>
                               | VALID FOR <valid_for_list>
                               | CONSTRAINT '<constraint_param>'
```

Restrictions to which properties apply to which statistic types are noted in the property descriptions.

**REFRESH TYPE refresh_type**

Specifies the strategy for the data statistics object.

```
<refresh_type> ::= { AUTO | MANUAL | DEFAULT }
```

AUTO specifies that the data statistics object is refreshed automatically when underlying data changes. AUTO is only supported on column store, extended store, and multistore tables.

MANUAL specifies that the database statistics object is not refreshed until a rebuild is explicitly requested by a REFRESH STATISTICS statement.

DEFAULT specifies that the database server decides the best refresh strategy based on the data source. For example, for data statistics objects on column store data sources, the database server applies AUTO for the default.

REFRESH TYPE only affects data statistics objects that are enabled.

**ENABLE on_off**

Controls whether the optimizer uses the data statistics object.

```
<on_off> ::= { ON | OFF }
```

ENABLE ON enables the optimizer to see the data statistics object. The data statistics object must be populated with data for the optimizer to use it. ENABLE ON specified with NO INITIAL REFRESH returns an error.

ENABLE ON is the default behavior.

ENABLE OFF disables the use of the data statistics object by the optimizer and prevents the ability to refresh the data statistics object. Data statistics objects that are not enabled can still be dropped. To make a data statistics object with ENABLE OFF accessible to the optimizer, execute an ALTER STATISTICS...ENABLE ON statement.

**BUCKETS unsigned_integer**
The BUCKETS property is only for use with TYPE HISTOGRAM or TOPK. For HISTOGRAM, BUCKETS specifies the maximum number of data buckets in the HISTOGRAM. For TOPK, BUCKETS specifies the K value.

The default is automatically determined by the data statistics building algorithm in use.

If a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

For column store, extended store, and multistore tables only, if a very small number of buckets is specified for a QOPTIMAL HISTOGRAM, then the algorithm may fail to build a valid HISTOGRAM either during the first build or during a subsequent refresh executed for the HISTOGRAM.

**QERROR numeric_literal**

Specifies the Q error to use for the q-optimal HISTOGRAM. You can specify this parameter when TYPE is HISTOGRAM and CONSTRAINT is QOPTIMAL. The default is automatically determined by the HISTOGRAM algorithm in use.

**QTHETA unsigned_integer**

Specifies a lower bound on the frequencies for which a q error constraint is applied for a q-optimal HISTOGRAM. You can specify this parameter when TYPE is HISTOGRAM and CONSTRAINT is QOPTIMAL. The default is automatically determined by the HISTOGRAM algorithm in use.

**MEMORY memory_bytes**

Specifies the maximum amount of memory, in bytes, to use for QOPTIMAL HISTOGRAMS.

\[
\text{memory_bytes} ::= \text{unsigned_integer}
\]

The MEMORY parameter limits the memory for QOPTIMAL HISTOGRAMS. MEMORY applies only to the QOPTIMAL HISTOGRAM algorithm. Small values for MEMORY may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

**MEMORY PERCENT memoryPercentage**

Specifies the maximum amount of memory to use for the data statistics object, expressed as a percentage of the space used by the data source.

\[
\text{memoryPercentage} ::= \text{unsigned_integer}
\]

HISTOGRAMS can use a large amount of memory for some data sources. \(\text{memoryPercentage}\) represents the maximum amount of memory that can be used for the data statistics object. For example, if a data source is a table column that uses 100 MB of memory, and \(\text{memoryPercentage}\) is 5, then the data statistics object for this column can use, at most, 5 MB for its in-memory representation.

The default is automatically determined by the HISTOGRAM algorithm in use. Small values for MEMORY PERCENT may cause the QOPTIMAL HISTOGRAM algorithm to pick a small number of buckets, which can lead to failures building or refreshing the HISTOGRAM.

**PERSISTENT on_off**
Specifies whether data statistics object data persists in the storage of the table, and only applies to QOPTIMAL HISTOGRAMS on column store tables. The default is PERSISTENT ON.

```plaintext
<on_off> ::= { ON | OFF }
```

Other statistics types are always persistent.

**ACCURACY numeric_literal**

Controls the time and space requirements to use for the SKETCH algorithms. This parameter can only be specified when TYPE is SKETCH and must be a number between 0 and 1, with larger values causing decreased time and space requirements but poorer SKETCH resolution. The default is 0.1.

**PREFIX BITS unsigned_integer**

Controls the number of bits the SKETCH algorithms use when constructing the SKETCH statistics. Specify this parameter when TYPE is SKETCH. Its value is an integer between 0 and 63. The default is 8.

**VALID FOR valid_for_list**

Defines how the data statistics object may be used. The VALID FOR clause is only permitted with statistics type SIMPLE.

```plaintext
<valid_for_list> ::= <usage>[, <usage>[, ...]]
<usage> ::= { ESTIMATION | DATA DEPENDENCY }
```

ESTIMATION initializes the data statistics object for use by the optimizer to improve selectivity estimation. SIMPLE data statistics objects are initialized for estimation use by default.

DATA DEPENDENCY applies to partitioned column store and multistore tables only. It initializes the data statistics object to be used by features that require higher (or more) data consistency (including automatically refreshing and rebuilding when needed), such as the dynamic partition pruning feature. For more information about the dynamic partition pruning feature, including the types of columns that can have data statistics defined for dynamic partition pruning, see the SAP HANA Administration Guide.

**CONSTRAINT**

Specifies constraints to use for the specified `<data_statistics_type>`.

- For HISTOGRAM, `<constraint_param>` specifies the mathematical constraint for the HISTOGRAM:

```plaintext
<constraint_param> ::= { QOPTIMAL | MAXDIFF }
```

QOPTIMAL HISTOGRAMS are for column store tables only. MAXDIFF HISTOGRAMS are only for row tables, virtual tables, extended tables, and extended partitions of multistore tables. The defaults are QOPTIMAL for column tables and MAXDIFF for other data sources. A non-default CONSTRAINT for HISTOGRAMS results in an error. HISTOGRAM sizing restrictions (BUCKETS, MEMORY, and MEMORY PERCENT) are applied per HISTOGRAM.

- For multistore tables, MAXDIFF is the default for the extended storage portion and QOPTIMAL is the default for the default storage portion.

- Histogram size limits apply separately to the HISTOGRAMS on the default storage partitions and the extended storage portion.
• For SKETCH, `<constraint_param>` specifies the algorithm to use to build the SKETCH. The default is LOGLOGCOUNTING; the remaining algorithms are for internal use.

```
<constraint_param> ::= KMINVAL | PCSA | LINEARCOUNTING | LOGCOUNTING | LOGLOGCOUNTING | SUPERLOGLOGCOUNTING
```

**initial_refresh**

Specifies whether to populate the data statistics object with data after creation.

```
<initial_refresh> ::= [ NO ] INITIAL REFRESH
```

 INITIAL REFRESH

Creates the definition of the data statistics object and populates it with data. The default behavior is INITIAL REFRESH.

 NO INITIAL REFRESH

Creates the definition of the data statistics object, but does not populate it with data.

Use NO INITIAL REFRESH when you want to change the underlying data before refreshing the data statistics object.

You cannot specify NO INITIAL REFRESH if ENABLE OFF is not specified.

**Permissions**

One of the following is true with regards to required permission:

• You own the table you are creating statistics on.
• You have the ALTER privilege on the object you are creating statistics on.

**Description**

The CREATE STATISTICS statement creates data statistics objects, which approximate the specified `<data_sources>`. The optimizer uses data statistics objects to estimate the properties of the data without directly accessing the data itself. They can be useful during query optimization and query execution.
Example

Create a q-optimal HISTOGRAM for each partition of the HANA column store, with a maximum 1000 buckets on the column T(X), where T is a multistore table. The QERROR that is not specified will be determined by the HISTOGRAM build algorithm.

```
CREATE STATISTICS ON MYSYSTEM.T(X) FOR DEFAULT STORAGE TYPE HISTOGRAM BUCKETS 1000
```

Create a HISTOGRAM for each storage part of the column T(X), where T is a multistore table. The HISTOGRAM build algorithm determines the algorithms to use for each part, as well as unspecified parameters -- BUCKETS for both parts and QERROR, QTHETA for the column store part. The statement creates two HISTOGRAMS.

```
CREATE STATISTICS ON MYSYSTEM.T(X) TYPE HISTOGRAM
```

Create a 10 bucket HISTOGRAM over the extended storage partitions of column T(COL2), where T is a multistore table.

```
CREATE STATISTICS "HISTOGRAM_T_COL2" ON MYSYSTEM.T(COL2) FOR EXTENDED STORAGE TYPE HISTOGRAM BUCKETS 10
```

Create a SKETCH over the column store partitions of T(COL2) and T(COL3), where T is a multistore table. This statement creates a single object representing a SKETCH over the pair of columns. Specify column lists within a single set of parentheses.

```
CREATE STATISTICS "SKETCH_T_COL2_COL3" ON MYSYSTEM.T(COL2,COL3) FOR DEFAULT STORAGE TYPE SKETCH
```

Create a SKETCH over columns T(COL2), T(COL3), and T(COL4), where T is a multistore table. This statement creates two objects: one for the default storage partitions of table T, and the other for the extended storage partitions. Each object represents a SKETCH over the joint distribution of the three columns.

```
CREATE STATISTICS ON MYSYSTEM.T(COL2,COL3,COL4) TYPE SKETCH
```

The following example creates a data statistics object of type SIMPLE and enables it for use with dynamic partition pruning:

```
CREATE STATISTICS MYSIMPLESTAT ON MYSYSTEM.MYTABLE(COL1) VALID FOR DATA DEPENDENCY;
```

Related Information

ALTER STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1305]
REFRESH STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1441]
DROP STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1395]
M_DATA_STATISTICS System View [page 1851]
M_SYSTEM_DATA_STATISTICS System View [page 2189]
5.2.21 CREATE TABLE Statement (Extended Store Table)  
[Dynamic Tiering]

Creates a new extended store table in the extended storage.

Syntax

CREATE [ COLUMN ] TABLE <table_name>  
(<table_contents_source>, ...)  
<using_extended_storage_clause>

Syntax Elements

**table_name**

<table_name> ::= [<schema_name>.]<identifier>

<table_name> ::= <unicode_name>

**table_contents_source**

You can create a table by describing the table elements, see below.

<table_contents_source> ::=  (<table_element>, ...)  
| [(<column_name>, ...)]

**table_element**

Defines the table columns with associated column or table constraints.

<table_element> ::=  
<column_definition> [<column_constraint>]  
| <table_constraint>

**column_definition**

Defines the table columns with associated column or table constraints.

<column_definition> ::= <column_name> {<data_type>  
| <lob_data_type>} [<default_value_clause>]

**column_constraint**

See <column constraint>.

**table_constraint**

See <table constraint>. 
**column_name** Specifies the column name.

```
<column_name> ::= <identifier>
```

**data_type**

Specifies the column data types.

```
<data_type> ::= DATE | TIME | SECONDDATE | TIMESTAMP | TINYINT | SMALLINT | INTEGER | BIGINT | SMALLDECIMAL | DECIMAL | REAL | DOUBLE | VARCHAR | NVARCHAR | ALPHANUM | SHORTTEXT | VARBINARY | BINTEXT
```

**lob_data_type**

Specifies the LOB data type.

```
<lob_data_type> ::= <lob_type_name> [MEMORY THRESHOLD memory_threshold_value]

<lob_type_name> ::= BLOB | CLOB | NCLOB
```

**default_value_clause**

Specifies a value to be assigned to the column if an INSERT statement does not provide a value for the column.

```
<default_value_clause> ::= DEFAULT <default_value_exp>

<default_value_exp> ::= NULL | <string_literal> | <signed_numeric_literal> | <unsigned_numeric_literal> | <datetime_value_function>

<datetime_value_function> ::= CURRENT_DATE | CURRENT_TIME | CURRENT_TIMESTAMP
```

**column_constraint**

Specifies the column constraint rules.

```
<column_constraint> ::= NULL
```
NOT NULL

Allows NULL values in the column. If NULL is specified it is not considered a constraint, it represents that a column that may contain a null value. The default is NULL.

NOT NULL

Prohibits NULL values in the column.

unique specification

Specifies unique constraints.

<unique specification> ::= UNIQUE [<unique_tree_type_index>] |
                        PRIMARY KEY [<unique_tree_type_index>]

If the index type is omitted, the SAP HANA database chooses the appropriate index by considering the column data type. If the index type is not specified, the SAP HANA database automatically selects an index type as follows:

<table>
<thead>
<tr>
<th>Index type</th>
<th>Criteria</th>
</tr>
</thead>
</table>
| CPBTREE    | • character string types
           | • binary string types
           | • decimal types
           | • when the constraint is a composite key
           | • when the constraint is a non-unique constraint |
| BTREE      | All other cases than specified for CPBTREE |

UNIQUE

Specifies a column as a unique key. A composite unique key enables the specification of multiple columns as a unique key. With a unique constraint, multiple rows cannot have the same value in the same column.

unique_tree_type_index

Specifies the index type.

<unique_tree_type_index> ::= { BTREE | CPBTREE }
Specifies a primary key constraint, which is a combination of a NOT NULL constraint and a UNIQUE constraint.

**table_constraint**

```plaintext
<table_constraint> ::=  
<unique_constraint_definition>
```

**unique_constraint_definition**

The unique specification. See `<unique_specification>`.

```plaintext
<unique_constraint_definition> ::=  
<unique_specification> (<unique_column_name_list>)
```

**unique_column_name_list**

Specifies the unique column name list, which can have one or more column names.

```plaintext
<unique_column_name_list> ::=  
<unique_column_name>[{,<
<unique_column_name>}...]
```

**unique_column_name**

A column name identifier.

```plaintext
<unique_column_name> ::=  
<identifier>
```

**using_extended_storage_clause**

Creates an extended store table.

```plaintext
(using_extended_storage_clause) ::=  
USING EXTENDED STORAGE [(ENABLE | DISABLE) DELTA]
```

---

**i Note**

- Extended storage must already exist.
- DBO, SYS, and PUBLIC are reserved words within extended storage. Schemas with these names cannot have extended store tables.

**ENABLE | DISABLE DELTA**

Enable row-level versioning (DELTAs) on the extended store table. Enabling DELTAs facilitates faster inserts.

---

**i Note**

DELTAs not supported for data types BLOB, CLOB and NCLOB.

---

**Permissions**

You have the EXTENDED STORAGE ADMIN privilege.
**Description**

The CREATE TABLE statement with the USING EXTENDED STORAGE clause creates an extended store table.

**Example**

You create table t1 as an extended store table where column A and B are INTEGER-type, column A is the primary key constraint, and DELTA is enabled.

```
CREATE COLUMN TABLE t1 (A INT PRIMARY KEY, B INT) USING EXTENDED STORAGE ENABLE DELTA;
```

You create table t2 as an extended store table where columns A, B, and C are INTEGER-type, columns A and B allow NULL values (default when not specified is NULL), but column C does not.

```
CREATE COLUMN TABLE t2 (A INT, B INT NULL, C INT NOT NULL) USING EXTENDED STORAGE;
```

You create table t3 as an extended store table where all columns are INTEGER-type, column A is the primary key and columns B and C have UNIQUE constraints.

```
CREATE COLUMN TABLE t3 (A INT PRIMARY KEY, B INT UNIQUE BTREE, C INT, UNIQUE CPBTREE(C)) USING EXTENDED STORAGE;
```

You create table t4 as an extended store table where columns A, B, and C are INTEGER-type, columns A and B do not allow NULL values, column C is the primary key and has a UNIQUE constraint.

```
CREATE COLUMN TABLE t4 (A INT NOT NULL, B INT NOT NULL, C INT, PRIMARY KEY CPBTREE(A, B), UNIQUE(C)) USING EXTENDED STORAGE;
```

**Related Information**

- TABLES System View [page 1669]
- TABLE_COLUMNS System View [page 1675]
- M_ES_TABLES System View [Dynamic Tiering] [page 2305]
- TABLE_COLUMNS System View [page 1675]
- TABLES System View [page 1669]
- M_ES_TABLES System View [Dynamic Tiering] [page 2305]
5.2.22 CREATE TABLE Statement (Multistore Table)  
[DYNAMIC TIERING]

Creates a new multistore table in the database.

Syntax

```
CREATE COLUMN TABLE <table_name>  
( <table_contents_source>)  
[ <logging_option> ]  
[ <auto_merge_option> ]  
[ <unload_priority_clause> ]  
[ <group_clause> ]  
[ <location_clause> ]  
[ <non-heterogeneous_partition_clause> ]  
[ <heterogeneous_partition_clause> ]
```

Syntax elements:

**Syntax Elements**

**table_name**

Specifies a name for the table

```
<table_name> ::= [<schema_name>.]<identifier>
```

**table_contents_source**

Specifies the source from which the table definition is derived.

```
<table_contents_source> ::= (<table_element>,...)
```

**table_element**

Defines the table columns with associated column or table constraints.

```
<table_element> ::= 
<column_definition> [ <column_constraint> ]  
| <table_constraint>
```

**column_definition**

Defines the table columns with associated column or table constraints.

```
<column_definition> ::= <column_name> { <data_type>      | <lob_data_type> } [ <default_value_clause> ] [ <col_gen_as_ident> ]  
<column_name> ::= <identifier>
```

**column_name**

Specifies the column name.

```
<column_name> ::= <identifier>
```
**data_type**

Specifies the column data types.

```
<data_type> ::= DATE
    | TIME
    | SECONDDATE
    | TINYINT
    | SMALLINT
    | INTEGER
    | BIGINT
    | DECIMAL [, ]
    | REAL
    | DOUBLE
    | VARCHAR [ (<unsigned_integer>) ]
    | NVARCHAR [ (<unsigned_integer>) ]
    | VARBINARY [ (<unsigned_integer>) ]
```

For tables with time-selection partitioning, the data type for the time-selection column must be NVARCHAR(8).

**lob_data_type**

Specifies the LOB data type.

```
<lob_data_type> ::= BLOB
    | CLOB
    | NCLOB
```

**default_value_clause**

Specifies a value to be assigned to the column if an INSERT statement does not provide a value for the column.

```
<default_value_clause> ::= DEFAULT <default_value_exp>
```

```
<default_value_exp> ::= NULL
    | <string_literal>
    | <signed_numeric_literal>
    | <unsigned_numeric_literal>
    | <datetime_value_function>
    | <string_value_function>
```

```
<datetime_value_function> ::= CURRENT_DATE
    | CURRENT_TIME
    | CURRENT_TIMESTAMP
    | CURRENT_UTCDATE
    | CURRENT_UTCTIME
    | CURRENT_UTCTIMESTAMP
```

**col_gen_as_ident**

Specifies an identity column in a multistore table.

```
<col_gen_as_ident> ::= GENERATED BY DEFAULT AS IDENTITY [(<sequence_option>)]
```

```
<sequence_option> ::= 
```
AS IDENTITY <sequence_option> generates ID values according to the sequence option.

**column_constraint**

Specifies the column constraint rules.

- **NULL**
  Allows NULL values in the column. If NULL is specified it is not considered a constraint, it represents that a column that may contain a null value. The default is NULL.

- **NOT NULL**
  Prohibits NULL values in the column.

**unique_specification**

Specifies unique constraints.

<table>
<thead>
<tr>
<th>Index type</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPBTREE</td>
<td>• character string types</td>
</tr>
<tr>
<td></td>
<td>• binary string types</td>
</tr>
<tr>
<td></td>
<td>• decimal types</td>
</tr>
<tr>
<td></td>
<td>• when the constraint is a composite key</td>
</tr>
<tr>
<td></td>
<td>• when the constraint is a non-unique constraint</td>
</tr>
</tbody>
</table>
UNIQUE

Specifies a column as a unique key. A composite unique key enables the specification of multiple columns as a unique key. With a unique constraint, multiple rows cannot have the same value in the same column.

A column specified as UNIQUE can contain multiple NULL values.

**unique_tree_type_index**

Specifies the index type.

```plaintext
<unique_tree_type_index> ::= { BTREE | CPBTREE }
```

**BTREE**

Specifies a B+-tree index. B+-tree is a tree that maintains sorted data, which performs efficient insertion, deletion, and search of records.

**CPBTREE**

Specifies a CPB+-tree index. CPB+-tree stands for Compressed Prefix B+-Tree, which is based on pkB-tree. CPB+-tree is a very small index because it uses 'partial key' that is only part of full key in index nodes. CPB+-tree shows better performance than B+-Tree for larger keys.

**PRIMARY KEY**

Specifies a primary key constraint, which is a combination of a NOT NULL constraint and a UNIQUE constraint.

**unique_inverted_type_index**

**Note**

The `<unique_inverted_type_index>` parameter is only supported on column store partitions.

Specifies the inverted index type.

```plaintext
<unique_inverted_type_index> ::= INVERTED [<composite_type>]
```

```plaintext
<composite_type> ::= HASH | VALUE
```

**INVERTED HASH**

Specifies that a hash should be used to encode the composite key in a condensed manner. This allows for faster equality queries over the composite keys as well as reduced memory requirements for storage of the composite key.

HASH should not be used as a composite type in cases where range queries or similarity queries on the composite keys are a significant part of the workload. In these cases VALUE or VALUE COMPRESSED should be used instead.
INVERTED VALUE
VALUE is the default composite key type.

table_constraint
The table constraint is a unique constraint.

```
<table_constraint> ::= <unique_constraint_definition>
```

unique_constraint_definition

```
<unique_constraint_definition> ::= 
    <unique_specification> (<unique_column_name_list>)
```

unique_column_name_list
Specifies the unique column name list, which can have one or more column names.

```
<unique_column_name_list> ::= <unique_column_name>     
    [{, <;unique_column_name}>...]
```

unique_column_name
A column name identifier.

```
<unique_column_name> ::=      <identifier>
```

logging_option

i Note
The `<logging_option>` parameter is only supported on column store partitions.

Specifies that table logging is activated for the created table.

```
<logging_option> ::= LOGGING
```

LOGGING
Specifies that table logging is activated. This is the default logging option.

auto_merge_option

i Note
The `<auto_merge_option>` parameter is only supported on column store partitions.

Specifies that automatic delta merge is triggered; this is the default. NO AUTO MERGE disables automatic
delta merging.

```
<auto_merge_option> ::= [ NO ] AUTO MERGE
```

unload_priority_clause
Specifies that priority of table to be unloaded from memory.

```
<unload_priority_clause> ::= 
    UNLOAD PRIORITY <unload_priority>
```
**unload_priority** is a digit between, and including, 0 and 9, where 0 means not-unloadable and 9 means earliest unload.

**group_clause**

Sets the group type, subtype, and name.

\[
\text{<group_clause> ::= <group> [ <group> ...]}\]

\[
\text{<group> ::= GROUP TYPE <identifier> | GROUP SUBTYPE <identifier> | GROUP NAME <identifier>}
\]

Grouping applies only to column store partitions within a multistore table.

**location_clause**

Specifies the index servers on which the column store and default store partitions will be created.

\[
\text{<location_clause> ::= AT [LOCATION] ( <indexserver_host_port> | ( <indexserver_host_port>, ...))}
\]

\[
\text{<indexserver_host_port> ::= 'hostname:<port_number>'}
\]

It is possible to determine the index servers on which the partitions are created. If you specify the LOCATION, the partitions will be created on these instances using round robin scheme. Duplicates in the list will be removed. If you specify the same number of instances as first level partitions in the partition specification, then each partition will be assigned to the respective instance in the list. All index servers in the list have to belong to the same instance.

Second level range partitions can reside in either default or extended storage. Only column store partitions reside on the indexserver. For a given first level partition, all second level column store partitions must reside on the same indexserver.

If no locations are specified, the partitions are determined by the system. If the number of partitions matches the number of servers — for example by using GET_NUM_SERVERS() — multiple CREATE TABLE calls will be distributed across all servers evenly. In case of a multi-level partitioning, this applies for the number of partitions of the first level.

This mechanism is useful if several tables are to be created which have a semantic relationship to each other. A table can be created in the specified location with <hostname>:<port_number>. <location_clause> can be specified when creating partitioned tables that are distributed on multiple instances.

**non-heterogeneous_partition_clause**

Partitions a table using a range, range-range, or hash-range partitioning scheme. A multistore table must have at least two partition ranges, one on the HANA host and one on the dynamic tiering host. These ranges must exist in the second-level range partition.

\[
\text{<non-heterogeneous_partition_clause> ::= PARTITION BY <range> | <range_range> | <hash_range> [ WITH PARTITIONING ON ANY COLUMNS { ON | OFF } ] [ WITH CROSS STORAGE UNIQUE CONSTRAINTS { ON | OFF } ] [ FOR EXTENDED STORAGE ENABLE DELTA ] [ FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS {ON | OFF} ]}
\]
Partitioning by Date values requires the format 'YYYYMMDD', 'YYYY-MM-DD', or 'YYYY/MM/DD'. Date is the most granular datetime value that can be used for partitioning. For more information about partitioning, see Table Partitioning in the SAP HANA Administration Guide.

range

Partitions the table using a single level range partitioning scheme.

```sql
<range> ::= RANGE ( <partition_expression> )
          ( USING DEFAULT STORAGE ( [ <time_selection>, ] [ <range_partition> ]
            USING EXTENDED STORAGE ( <range_partition> ) )
```

range_range

Partitions the table using a range-range partitioning scheme.

```sql
<range_range> ::= RANGE ( <partition_expression> ) ( <range_partition> ),
                 RANGE( <partition_expression> )
                 ( USING DEFAULT STORAGE ( [ <time_selection>, ] [ <range_partition> ]
                 USING EXTENDED STORAGE ( <range_partition> ) )
```

hash_range

Partitions the table using a hash-range partitioning scheme.

```sql
<hash_range> ::= HASH ( <partition_expression> ) PARTITION {
          <num_partitions> | GET_NUM_SERVERS() },
          RANGE( <partition_expression> )
          ( USING DEFAULT STORAGE ( [ <time_selection>, ] [ <range_partition> ]
          USING EXTENDED STORAGE ( <range_partition> ) )
```

GET_NUM_SERVERS() Returns the number of servers/partitions according to Table Placement. See Table Placement in the SAP HANA Administration Guide.

num_partitions

The number of hash partitions to be created for the table.

```sql
<num_partitions> ::= <unsigned_integer>
```

partition_expression

Declares the specifier that segregates data into partitions.

```sql
<partition_expression> ::= <column_name>
                      | YEAR(<column_name>)
                      | MONTH(<column_name>)
```

column_name Specifies the partitioning column. The column must be of a data type supported for the partitioning scheme.

Hash partitions

TINYINT, SMALLINT, INT, BIGINT, DECIMAL, DECIMAL(p,s), CLOB, NCLOB, SHORTTEXT, VARCHAR, NVARCHAR, BLOB, VARBINARY, DATE, TIME, TIMESTAMP and SECONDDATE. Memory LOBs (ST_MEMORY_LOB type only) are supported but not disk LOBs.
Range partitions STRING, TINYINT, SMALLINT, INT, SHORTTEXT, VARCHAR, NVARCHAR, DATE, TIME, TIMESTAMP, SECONDDATE, FIXED, RAW (SQL Binary/Varbinary). Memory LOBs (ST_MEMORY_LOB type) are supported but not disk LOBs.

YEAR / MONTH Specifies the precision of the date based partitioning column.

range_partition
Specifies the ranges of the first- or second-level range partition.

<range_partition> ::= <part_range> [ , <part_range> [ , ... ] ]

part_range
Specifies the values and properties for a first or second-level range partition.

<part_range> ::= PARTITION <range_values> [ <range_prop_list> ] [ , PARTITION <range_values> [ <range_prop_list> [ , ... ] ]

range_values
Specifies the range of the partition.

<range_values> ::= <min_value> <= VALUES < <max_value> | VALUE[S] = <target_value> | PARTITION OTHERS

min_value
The lower value of a partition specifier. Value cannot be negative.

<min_value> ::= <string_literal> | <numeric_literal>

max_value
The upper value of a partition specifier. Value cannot be negative.

<max_value> ::= <string_literal> | <numeric_literal>

target_value
The target value of a single partition specifier. Value cannot be negative.

<target_value> ::= <string_literal> | <numeric_literal>

PARTITION OTHERS
Specifies that all other values that are not covered by the partition specification will be gathered into one partition. Can only be added to a first or second level default store partition.

time_selection
 Specifies the CURRENT partition for aging.

<time_selection> ::= VALUE = '00000000' IS CURRENT

CURRENT is required for tables with time selection partitioning. Time selection is supported for a range-range or hash-range partitioned table. The <data_type> of the time selection column must be NVARCHAR(8). The <time_selection> clause must:
• reside in the second-level for range-range or hash-range partition schemes
• be defined as the first partition on the second level range in the default store
• be a single value partition with the value 00000000

<time_selection> partitioning also requires the WITH PARTITIONING ON ANY COLUMNS clause set to ON.

range_prop_list

Specifies the properties for the partition.

```
<range_prop_list> ::= <range_prop> [ <range_prop> ... ] ]
<range_prop> ::= { <persistent_memory_spec_clause> | <numa_node_preference_clause> }
```

Range properties can only be defined for the default storage in the second-level range partition.

**persistent_memory_spec**

Enables or disables persistent memory storage preference at the table, range partition, or column level, depending on where the clause is situated in the CREATE statement. For example, when specified inside the `<column_definition>` clause, it enables or disables persistent memory storage for the column.

```
<persistent_memory_spec> ::= PERSISTENT MEMORY <pm_preference>
/pm_preference> ::= { ON | OFF }
```

**numa_node_preference_clause**

Sets the NUMA node preferences. Although this clause is defined here at the table level, `<numa_node_preference_clause>` can be set in various locations such as range partition definitions (not hash or round-robin) and column definitions, as indicated in the syntax within the topic. `<numa_node_preference_clause>` is not supported for heterogeneous partitions.

```
<numa_node_preference_clause> ::= NUMA NODE { ( <numa_node_index_spec> ) }<numa_node_index_spec> ::= <integer_const> ... <numa_node_spec> ::= ( <integer_const> )<range_node_spec> ::= <integer_const> TO <integer_const>
```

**integer_const**

`<integer_const>` cannot be a negative number.

**numa_node_spec**

Specify one or more single NUMA nodes (`<single_node_spec>`), or one or more NUMA node ranges (`<range_node_spec>`), or a mixture of both.

NUMA node indexes should be specified in the range of 0 to one less than max_numa_node_count, where max_numa_node_count is the number of NUMA nodes configured for the system. If the NUMA node index specified is greater than or equal to max_numa_node_count, then a random NUMA node in the range of 0 to one less than max_numa_node_count is selected for allocation. For example, on a system where max_numa_node_count is set to 8, if you specify a NUMA node index of 10, then any node in the range of 0 to 7 (inclusive) is chosen randomly for allocation.

**heterogeneous_partition_clause**
Partitions the table by using a one or two level heterogeneous range partitioning scheme. Only the range partition type is supported. A multistore table must have at least two partition ranges, one on the HANA host and one on the dynamic tiering host.

```
<heterogeneous_partition_clause> ::=  
PARTITION BY RANGE ( <partition_expression> )   
  ( ( <part_spec> <HANA_loc> )   
    [ SUBPARTITION BY RANGE ( <partition_expression> ) ( <part_spec>   
  ) ],   
  ( <part_spec> <ES_loc> )   
    [ SUBPARTITION BY RANGE ( <partition_expression> ) ( <part_spec>   
  ) ] )   
  [ PRIMARY KEY UPDATE { ON | OFF } ]  
<part_spec> ::=   
<part_range> [, <part_range> [,...] ]
```

**part_range**

Specifies the range specifier for the first-level heterogeneous range partition.

```
<part_range> ::=  
PARTITION <range_values> [INSERT { OFF | ON } ] [   
<numa_node_preference_clause> ]<>
```

**range_values**

Specifies the range of the first or second level heterogeneous range partition.

```
<range_values> ::=       <min_value> <= VALUES < <max_value>      | VALUE[S] = <target_value>      | OTHERS
```

**min_value**

The lower value of a partition specifier. Value cannot be negative.

```
<min_value> ::= <string_literal>      | <numeric_literal>
```

**max_value**

The upper value of a partition specifier. Value cannot be negative.

```
<max_value> ::= <string_literal>      | <numeric_literal>
```

**target_value**

The target value of a single partition specifier. Value cannot be negative.

```
<target_value> ::= <string_literal>      | <numeric_literal>
```

**OTHERS**

Specifies that all other values that are not covered by the partition specification will be gathered into one partition. OTHERS property is not supported on partitions in extended storage.

**HANA_loc**
Specifies where the default storage partition resides. If a location is not specified for a partition, the default places the partition on the indexserver in the default store.

\[<\text{HANA\textunderscore loc}>::\{\text{USING DEFAULT STORAGE} | \text{AT LOCATION} '<\text{HANA\_host}>:<\text{HANA\_port}>'\}\]

\textbf{ES\_loc}

Specifies where the extended storage partition resides. If a location is not specified for a partition, the default places the partition on the indexserver in the default store.

\[<\text{ES\_loc}>::\{\text{USING EXTENDED STORAGE} | \text{AT LOCATION} '<\text{ES\_host}>:<\text{ES\_port}>'\}\]

\textbf{INSERT} {\textbf{ON} | \textbf{OFF}}

(Appplies to heterogeneous range partitions only) Specifies if INSERT statements are allowed on a partition. The INSERT parameter can be defined on a first or second level partition. If defined at the first level, it applies to all second level partitions within the first level partition. If defined at both the first and second level, any value at the second level overrides the first level value. If not specified, default is ON.

\textbf{PRIMARY KEY} {\textbf{ON} | \textbf{OFF}}

(Appplies to heterogeneous range partitions only) Specifies if UPDATE statements are allowed on primary key columns. If not specified, default is ON.

\textbf{WITH PARTITIONING ON ANY COLUMNS}

Required when table uses time selection, set to ON. The partitioning column must be a subset of the primary key, if defined. If not specified, default (OFF) is applied.

\textbf{WITH CROSS STORAGE UNIQUE CONSTRAINTS}

Not supported for multistore tables. If not specified, default (OFF) is applied.

\textbf{FOR EXTENDED STORAGE ENABLE DELTA}

Specifies whether extended storage is configured for enable row-level versioning.

\textbf{Note}

Delta not supported for data types BLOB, CLOB and NCLOB.

\textbf{FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS}

Required for multistore tables using time selection partitioning, set to ON. Set to OFF (default) for all other multistore tables. If not specified, default is applied.

\textbf{FOR DEFAULT STORAGE} {\textbf{ALL} | \textbf{NON CURRENT}} \textbf{PARTITIONS} \{\textbf{PAGE} | \textbf{COLUMN}\} \textbf{LOADABLE}

\textbf{PAGE LOADABLE} is only supported for NON CURRENT partitions in a multistore table using time selection.

\textbf{Permissions}

This statement requires the EXTENDED STORAGE ADMIN privilege.

\textbf{Description}

The CREATE TABLE statement creates a new multistore table in the database.
Delta can’t be enabled when creating a multistore table. Use the ALTER TABLE command to enable it after the fact.

When creating a multistore table, refer to the section Create a Multistore Table in the SAP HANA Dynamic Tiering Administration Guide for details and specific examples on supported clause behaviors.

**Examples**

This example creates a range partitioned multistore table without time-selection partitioning:

```
CREATE COLUMN TABLE t1 (a INT, b NVARCHAR(8))
  PARTITION BY RANGE (a)
  (USING DEFAULT STORAGE (PARTITION 10 <= VALUES < 20, PARTITION VALUE = 25, PARTITION OTHERS)
   USING EXTENDED STORAGE (PARTITION 30 <= VALUES < 40, PARTITION VALUE < 50))
FOR EXTENDED STORAGE ENABLE DELTA
AT LOCATION ('hosta:30003', 'hostb:30003');
```

This example creates a range-range partitioned multistore table without time-selection partitioning:

```
CREATE COLUMN TABLE t2 (a INT, b INT)
  PARTITION BY RANGE (a)
  (PARTITION 5 <= VALUES < 10),
  RANGE (b)
  (USING DEFAULT STORAGE (PARTITION 10 <= VALUES < 20, PARTITION VALUE = 25, PARTITION OTHERS)
   USING EXTENDED STORAGE (PARTITION 30 <= VALUES < 40, PARTITION VALUE < 50))
FOR EXTENDED STORAGE ENABLE DELTA
AT LOCATION ('hosta:30003', 'hostb:30003');
```

This example creates a hash-range partitioned multistore table without time selection partitioning:

```
CREATE COLUMN TABLE t3 (a INT, b INT)
  PARTITION BY HASH (a) PARTITIONS 5, RANGE (b)
  (USING DEFAULT STORAGE (PARTITION 10 <= VALUES < 20, PARTITION VALUE = 25, PARTITION OTHERS)
   USING EXTENDED STORAGE (PARTITION 30 <= VALUES < 40, PARTITION VALUE < 50))
FOR EXTENDED STORAGE ENABLE DELTA
AT LOCATION ('hosta:30003', 'hostb:30003');
```

This example creates a range partitioned multistore table with time selection partitioning:

```
CREATE COLUMN TABLE ts1 (a INT, b INT, c NVARCHAR(8))
  PARTITION BY RANGE (c)
  (USING DEFAULT STORAGE (PARTITION value = '00000000' IS CURRENT, PARTITION OTHERS,
   PARTITION '20150101' <= VALUES < '20160101',
   PARTITION '20140101' <= VALUES < '20150101')
  USING EXTENDED STORAGE (PARTITION '20130101' <= VALUES < '20140101',
   PARTITION '20120101' <= VALUES < '20130101'))
FOR EXTENDED STORAGE ENABLE DELTA
WITH PARTITIONING ON ANY COLUMNS ON
FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS OFF
FOR DEFAULT STORAGE NON CURRENT PARTITIONS PAGE LOADABLE
AT LOCATION ('hosta:30003', 'hostb:30003');
```
This example creates a range-range partitioned multistore table with time selection partitioning:

```
CREATE COLUMN TABLE ts2 (a INT, b INT, c NVARCHAR(8), PRIMARY KEY (a, c))
PARTITION BY RANGE (a)
(PARTITION 10 <= VALUES < 20, PARTITION 20 <= VALUES < 30),
RANGE (c)
(USING DEFAULT STORAGE (PARTITION value = '00000000' IS CURRENT, PARTITION OTHERS,
  PARTITION '20150101' <= VALUES < '20160101',
  PARTITION '20140101' <= VALUES < '20150101')
USING EXTENDED STORAGE (PARTITION '20130101' <= VALUES < '20140101',
  PARTITION '20120101' <= VALUES < '20130101'))
FOR EXTENDED STORAGE ENABLE DELTA
WITH PARTITIONING ON ANY COLUMNS ON
FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS OFF
FOR DEFAULT STORAGE NON CURRENT PARTITIONS PAGE LOADABLE
AT LOCATION ('hosta:30003', 'hostb:30003');
```

This example creates a hash-range partitioned multistore table with time selection partitioning:

```
CREATE COLUMN TABLE ts3 (a INT, b INT, c NVARCHAR(8), PRIMARY KEY (a, c))
PARTITION BY HASH (a) PARTITIONS 5, RANGE (c)
(USING DEFAULT STORAGE (PARTITION value = '00000000' IS CURRENT, PARTITION OTHERS,
  PARTITION '20150101' <= VALUES < '20160101',
  PARTITION '20140101' <= VALUES < '20150101')
USING EXTENDED STORAGE (PARTITION '20130101' <= VALUES < '20140101',
  PARTITION '20120101' <= VALUES < '20130101'))
FOR EXTENDED STORAGE ENABLE DELTA
WITH PARTITIONING ON ANY COLUMNS ON
FOR NON CURRENT PARTITIONS UNIQUE CONSTRAINTS OFF
FOR DEFAULT STORAGE NON CURRENT PARTITIONS PAGE LOADABLE
AT LOCATION ('hosta:30003', 'hostb:30003');
```

This example creates a heterogeneous range partitioned multistore table. Since no location is specified for the 10 to 20 range, it resides in default storage. The USING property applies only to the range immediately preceding it. Since no location is specified for range 40 to 50, it resides in default storage while 60 to 70 resides in extended storage. Though ranges 30 to 40, 50 to 60, and 60 to 70 reside in different stores or servers, they each have a subpartition with the same range (100 to 150).

```
CREATE COLUMN TABLE P3 (L INT, M INT) PARTITION BY RANGE (L)
  (( PARTITION 10 <= VALUES < 20, PARTITION 20 <= VALUES < 30 USING DEFAULT STORAGE)
  SUBPARTITION BY RANGE (M) (PARTITION 100 <= VALUES < 150, PARTITION VALUE = 200),
  ( PARTITION 30 <= VALUES < 40 AT LOCATION '<indexserver1>:<port>'
  PARTITION 40 <= VALUES < 50, PARTITION 60 <= VALUES < 70 USING EXTENDED STORAGE)
  SUBPARTITION BY RANGE (M) (PARTITION 100 <= VALUES < 150));
```

This example creates a table that has an identity column B, with a start value of 100 and increment of 10:

```
CREATE COLUMN TABLE t5 (A INT, B INT GENERATED AS IDENTITY (START WITH 100 INCREMENT BY 10))
PARTITION BY RANGE (A)
  ((PARTITION 10 <= VALUES < 20, PARTITION 20 <= VALUES < 30 USING DEFAULT STORAGE)
  SUBPARTITION BY RANGE (B) (PARTITION 100 <= VALUES < 150, PARTITION VALUE = 200),
  (PARTITION 30 <= VALUES < 40, PARTITION 40 <= VALUES < 50, PARTITION 60 <= VALUES < 70 USING EXTENDED STORAGE)
  SUBPARTITION BY RANGE (B) (PARTITION 100 <= VALUES < 150));
```
5.2.23 DROP EXTENDED STORAGE Statement [Dynamic Tiering]

Removes an existing extended storage configuration from the SAP HANA database.

Syntax

```
DROP EXTENDED STORAGE [<drop_option>]
```

Syntax Elements

**drop_option**

When `<drop_option>` is not specified a restrict drop is performed.

```
<drop_option> ::= CASCADE | RESTRICT
```

**CASCADE**

Drops the extended storage and dependent objects.

**RESTRICT**

Drops the extended storage only when dependent objects do not exist. If this drop option is used and a dependent object exists an error is thrown.
Permissions

You must have the EXTENDED STORAGE ADMIN privilege.

Description

The DROP EXTENDED STORAGE statement removes an existing extended storage configuration from the SAP HANA database.

Dropping extended storage drops all dynamic tiering partitions for multistore.

⚠️ Caution

Dropping extended storage after a full data backup, but before recovery, will break business continuity. See SAP Note 2375865 – Backup and Recovery Functional Restrictions.

Example

This drops an extended storage if there are no dependent objects like extended tables.

```
DROP EXTENDED STORAGE
```

The above example drops extended storage with all dependent objects like extended tables.

```
DROP EXTENDED STORAGE CASCADE
```

5.2.24 DROP STATISTICS Statement (Extended Store Table) [Dynamic Tiering]

Drops user-defined data statistics objects that the query optimizer uses to make decisions for query plans.

Syntax

```
DROP STATISTICS { <data_statistics_name>[,<data_statistics_name>[,...] ] 
| ON <data_sources> [ [HAVING] <match_properties> ] )
```
Syntax Element

**data_statistics_name**

Specifies the name of the data statistics object.

```
<data_statistics_name> ::= [<schema_name>].<identifier>
```

**data_sources**

Specifies the data source(s) of the data statistics objects.

```
<data_sources> ::=  
<table_name> [ ( <column_name>, <column_name>[,...] ) ] [ <match_type> ]
```

**match_properties**

Specifies properties to use for matching when selecting data statistics.

```
<match_properties> ::= <match_property>[...]
<match_property> ::=  TYPE <data_statistics_type>  |  REFRESH TYPE <refresh_type_filter>
```

If TYPE is not specified, then all data statistics objects of any type on the specified data sources are refreshed (ALL). For descriptions of the supported data statistics types see the CREATE STATISTICS Statement (Extended Store Table) topic.

**data_statistics_type**

Specifies the type of data statistics objects to match when selecting the data statistics.

```
<data_statistics_type> ::= TYPE <type_name>
```

```
<type_name> ::=  HISTOGRAM
              |  SIMPLE
              |  TOPK
              |  SKETCH
              |  SAMPLE
              |  RECORD COUNT
              |  ALL
```

**refresh_type_filter**

Specifies the refresh strategy to match in the data statistics objects when selecting data statistics. ALL is the default.

```
<refresh_type_filter> ::= AUTO  |  MANUAL  |  ALL
```

Permissions

One of the following is true:
You own the table you are creating statistics on.
You have the ALTER privilege on the object you are dropping.

**Example**

Drop all statistics over \( T(COL2) \) extended storage partitions.

```sql
DROP STATISTICS ON MYSHEMA.T(COL2) FOR EXTENDED STORAGE
```

Drop data statistics objects with the name "HISTOGRAM_T_X" or "HISTOGRAM_T_Y".

```sql
DROP STATISTICS "HISTOGRAM_T_X","HISTOGRAM_T_Y";
```

Drop the HISTOGRAMS on table columns \( T(A), T(B), \) and \( T(C) \):

```sql
DROP STATISTICS ON MYSHEMA.T(A,B,C) TYPE HISTOGRAM;
```

**Related Information**

CREATE STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1361]
ALTER STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1299]
REFRESH STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1437]
M_DATA_STATISTICS System View [page 1851]
M_SYSTEM_DATA_STATISTICS System View [page 2189]

5.2.25 DROP STATISTICS Statement (Multistore Table)
[Dynamic Tiering]

Drops user-defined data statistics objects that the query optimizer uses to make decisions for query plans.

**Syntax**

```sql
DROP STATISTICS { <data_statistics_name>[,.<data_statistics_name>[,...] ]
| ON <data_sources> [ [HAVING] <match_properties> ] }
```
Syntax Element

data_statistics_name

Specifies the name of the data statistics object.

\[
<\text{data_statistics_name}> ::= [<\text{schema_name}>.]<\text{identifier}>
\]

\[
<\text{schema_name}> ::= <\text{identifier}>
\]

data_sources

Specifies the data source(s) of the data statistics objects.

\[
<\text{data_sources}> ::= \begin{align*}
<\text{table_name}> \ [ ( \ <\text{column_name}>, <\text{column_name}>, \ldots ) \ ] \\
\text{FOR} \ \{ \ \text{DEFAULT} \ | \ \text{EXTENDED} \ \} \ \text{STORAGE} \ [ \ <\text{match_type}> \ ]
\end{align*}
\]

table_name

Specifies the table on which the data statistics are defined.

\[
<\text{table_name}> ::= [<\text{schema_name}>.]<\text{identifier}>
\]

column_name

Specifies the column for which the data statistics are defined.

\[
<\text{column_name}> ::= <\text{identifier}>
\]

If no <column_name> is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

FOR (DEFAULT | EXTENDED) STORAGE

The STORAGE clause can be specified only for multistore tables. Specify DEFAULT when the data statistics object is created on the column store part. Specify EXTENDED STORAGE when the data statistics object is created on the extended storage part. If you do not specify anything, then two data statistics objects are created, one on the extended storage part and one on the column storage part.

Statistics of some types can only be created on one storage class partition; for example, SKETCH on HANA column store partitions, and RECORD COUNT on extended storage partitions. Attempting to create such objects on a multistore table without specifying storage type returns an error.

You can implement different statistics types for different storage classes, and create statistics to address specific performance problems that only affect the query plan for one of the storage classes of the table. The statement returns an error if no partitions of the specified storage type exist.

c

match_type

Controls which data statistics objects to match to <data_sources>.

\[
<\text{match_type}> ::= \text{EXACT} \ | \ \text{CASCADE}
\]

If <match_type> is not specified, then any data statistics object(s) that reference all or some of the columns, but no other columns specified in <data_sources> are refreshed.
Specify **EXACT** to refresh data statistics objects that precisely match `<data_sources>` (including column order).

Specify **CASCADE** to refresh data statistics objects that reference at least one column in `<data_sources>`.

Use this table to understand how matching is performed based on `<match_type>` when `<data_sources>` is `T(A, B, C):

<table>
<thead>
<tr>
<th>Match type</th>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
</table>
| (not specified)     | T(A,C)          | T(A,X) - because T(X) is not a column in `<data_sources>`.
|                     | T(C)            |                     |
|                     | T(B,A)          |                     |
| **EXACT**           | T(A,B,C)        | T(B,A,C) - because the column order is different than the column order of `<data_sources>`.
|                     |                 | T(A) - because it does not contain the exact same columns and column order of `<data_sources>`.
|                     |                 | T(C,B,A,X) - because T(X) is not a column in `<data_sources>`.
| **CASCADE**         | T(A,C)          | T(X) - because it does not contain any columns that match the columns in `<data_sources>`.
|                     | T(C)            |                     |
|                     | T(B,A)          |                     |
|                     | T(A,B,C)        |                     |
|                     | T(A,X)          |                     |
|                     | T(B,C)          |                     |
|                     | T(A)            |                     |
|                     | T(C,B,A,X)      |                     |

**match_properties**

Specifies properties to use for matching when selecting data statistics.

```
<match_properties> ::= <match_property> [...]  
<match_property> ::= TYPE <data_statistics_type>  |  REFRESH TYPE <refresh_type_filter>
```

If `TYPE` is not specified, then all data statistics objects of any type on the specified data sources are altered (ALL). For descriptions of the supported data statistics types see the *CREATE STATISTICS Statement (Multistore Table)* topic.

**data_statistics_type**
Specifies the type of data statistics objects to match when selecting the data statistics.

```
<data_statistics_type> ::= TYPE <type_name>
[type_name] ::= HISTOGRAM | SIMPLE | TOPK | SKETCH | SAMPLE | RECORD COUNT | ALL
```

**refresh_type_filter**

Specifies the refresh strategy to match in the data statistics objects when selecting data statistics. ALL is the default.

```
<refresh_type_filter> ::= AUTO | MANUAL | ALL
```

**Permissions**

One of the following is true:

- You own the table you are creating statistics on.
- You have the ALTER privilege on the object you are dropping.

**Example**

Drop all statistics over T(COL2) extended storage partitions.

```
DROP STATISTICS ON MYSHEMA.T(COL2) FOR EXTENDED STORAGE
```

Drop data statistics objects with the name "HISTOGRAM_T_X" AND "HISTOGRAM_T_Y".

```
DROP STATISTICS MYSHEMA.HISTOGRAM_T_X,MYSHEMA.HISTOGRAM_T_Y;
```

Drop the HISTOGRAMS on table columns T(A), T(B), and T(C):

```
DROP STATISTICS ON MYSHEMA.T(A,B,C) TYPE HISTOGRAM;
```

Drop all histograms on table T, which drops any existing histograms over either partitions in the HANA column store or partitions in extended storage.

```
DROP STATISTICS ON T HAVING TYPE HISTOGRAM;
```

**Related Information**

CREATE STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1366]
5.2.26 EXPORT Statement (Extended Store Table) [Dynamic Tiering]

Exports extended store tables, views, column views, synonyms, sequences, and procedures in CSV format.

Syntax

```
EXPORT <export_object_name_list> AS CSV
    INTO <path> [ <archive_file_name> ]
    [ WITH <export_option_list> ]
    [ <query_export_specification> ]
```

Syntax Elements

**export_object_name_list**
Species a list of objects to export.

```
<export_object_name_list> ::=    <export_object_name>[, <export_object_name>[,...] ]   | { ALL | * }   
<export_object_name> ::= [ <schema_name>.]{<identifier> | * }   
```

**<schema_name> ::= <unicode_name>**

**<schema_name.>.<identifier>** specifies objects in a schema for export. Specify **<schema_name."*">** to select all objects within the specified schema for export. Specify ALL (without **<schema_name>.**) to select all objects from all schemas in the system for export.

**AS CSV**
The format to export the data to.

BINARY format is not supported. If specified, data is exported in CSV format with gzip compression

**INTO path [ archive_file_name ]**
Specifies the location where export files are placed.

```
<path> [ <archive_file_name> ] ::= <string_literal>
```
Include the file extension .tar.gz or .tgz on <archive_file_name> to export the data to an archive file.

File locations for extended storage must be on a shared file system available to both SAP HANA and the extended storage system.

For security reasons, the path must not contain symbolic links and must not point inside the database instance folder, except its backup and work subfolders. Examples of valid values include:

`'/tmp'`  
`'/hana/shared/HDB/HDB00/backup'`  
`'/hana/shared/HDB/HDB00/work/myexport.tgz'`

WITH export_option_list
A list of export options.

<export_option_list> ::= <export_option>[, <export_option>[,...] ]

<export_option> ::=  REPLACE  
| CATALOG ONLY  
| NO DEPENDENCIES   | THREADS <number_of_threads>  
| STATISTICS ONLY  
| NO STATISTICS  
| TRACE

REPLACE
If REPLACE is not specified, then an error is returned if previously exported data exists in the specified export directory.

CATALOG ONLY
Specifies to export only the database catalog.

NO DEPENDENCIES
Specifies to not export the underlying dependencies of an export object.

THREADS number_of_threads
Specifies the number of process threads to be used for concurrent export processing.

<number_of_threads> ::= <unsigned_integer>

The THREADS parameter specifies how many objects are exported in parallel, the default is 1. Increasing number of threads reduces export time, but can also negatively affect database system performance. Consider the following items when you use this parameter. When exporting:

- A single table – THREADS has no effect.
- A view or procedure – use two or more threads, up to the number of dependent objects.
- A whole schema – consider using more than 10 threads. With a maximum being the number of CPU cores in the system.
- A whole BW or ERP system database with tens of thousands of tables by using the ALL keyword – you can use up to 256 threads.

STATISTICS ONLY
Data statistics objects are exported by default, along with the objects they reference in `<export_object_name_list>` (for example, tables). Export behavior depends on what you specify. If you specify:

- STATISTICS ONLY (no CATALOG ONLY) – both the data and metadata for data statistics objects are exported, as well as the metadata for the objects in `<export_object_name_list>`.
- STATISTICS ONLY and CATALOG ONLY – only the metadata for the data statistics objects and the objects in `<export_object_name_list>` are exported; no data is exported.

**NO STATISTICS**

Excludes data statistics objects from the export. Metadata and data for the objects in `<export_object_name_list>` are still exported, and they are impacted as normal by whether CATALOG ONLY is specified.

**TRACE export_trace_file_name**

Writes execution information to the specified trace file when exporting data.

```
<export_trace_file_name> ::= <string_literal>
```

The info level trace file contains detailed error descriptions for troubleshooting. The file is saved to the current trace directory.

The contents of the trace file can be found in the M_TRACEFILE_CONTENTS system view and contains information on:

- The export list
- Dependencies to the export list
- Feasibility checks on the export
- Metadata exported
- The format of the exported data (CSV)
- The single archive for a single archive export
- The results table from the export

To delete the trace file, use the ALTER SYSTEM REMOVE TRACES statement.

**query_export_specification**

Specifies a query to use for the export. See the SAP HANA SQLScript Reference for information on Query Exports.

```
<query_export_specification> ::= 
ON <sqlscript_location_list> FOR <procedure_call_statement>
```

**Permissions**

You must have the EXPORT system privilege, and have SELECT privilege on the objects being exported.
Description

The EXPORT command exports tables, views, column views, synonyms, sequences, or procedures in CSV format.

You can only import exported binary files that were generated on an SAP HANA system with the same endianness.

Detailed results of the last execution of the EXPORT statement are stored in the session-local temporary table `<current_schema>#EXPORT_RESULT`. If no EXPORT has not been executed in the current session, selecting from `<current_schema>#EXPORT_RESULT` returns an error. If `<current_schema>` of the session is invalid, then EXPORT throws an exception as it cannot create its result table in `<current_schema>`.

### Column Table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(255)</td>
<td>Schema of the exported object.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(255)</td>
<td>Name of the exported object.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(16)</td>
<td>Type of the exported object (TABLE, VIEW, and so on).</td>
</tr>
<tr>
<td>LOCATION</td>
<td>VARCHAR(32)</td>
<td>Location (&lt;host&gt;:&lt;port&gt;) where the object was exported.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>Export status (done, skipped, failed).</td>
</tr>
<tr>
<td>ERROR</td>
<td>VARCHAR(512)</td>
<td>Error text in case of export failure.</td>
</tr>
</tbody>
</table>

For more information on permissions on the exported files, see SAP HANA Security Checklists and Recommendations.

Examples

Example 1: Export the information from MY_SCHEMA TAB1 in CSV format and overwrite data previously exported.

```
EXPORT MY_SCHEMA.TAB1 AS CSV INTO '/tmp' WITH REPLACE;
```

Example 2: Export all the database objects in schema A and B, replacing any existing export that may exist in the /tmp directory.

```
EXPORT A."*", B."*" AS CSV INTO '/tmp' WITH REPLACE THREADS 10;
```

Example 3: Export the results of a query where tables T1 and T2 are extended tables.

```
EXPORT ALL AS CSV INTO '/usr/sap/JL01/HDB00/work' FOR SELECT * FROM T1, T2;
```

Example 4: Save and export an entire directory in compressed form.

```
EXPORT MY_EXTENDED_TABLE AS CSV INTO '/usr/sap/P30/HDB30/work/myexport.tar.gz' WITH REPLACE;
```
Example 4: Export the schema MY_SCHEMA and create the trace file export.trc.

```sql
EXPORT MY_SCHEMA."*" INTO '/usr/sap/MS1/HDB01/work' WITH TRACE 'export.trc';
```

**Related Information**

ALTER SYSTEM CANCEL [WORK IN] SESSION Statement (System Management) [page 516]
M_CONNECTIONS System View [page 1785]
SAP HANA Security Checklists and Recommendations
M_TRACEFILE_CONTENTS System View [page 2247]
ALTER SYSTEM REMOVE TRACES Statement (System Management) [page 564]
IMPORT Statement (Extended Store Table) [Dynamic Tiering] [page 1409]

5.2.27 EXPORT Statement (Multistore Table) [Dynamic Tiering]

Exports multistore tables, views, column views, synonyms, sequences, and procedures in CSV format.

**Syntax**

```sql
EXPORT <export_object_name_list>
AS <export_format>
INTO <path> [ <archive_file_name> ]
[ WITH <export_option_list> ]
[ <query_export_specification> ]
```

**Syntax Elements**

**export_object_name_list**

Specifies a list of objects to export. All and * options are only used to export all objects from a specified schema.

```sql
<export_object_name_list> ::= <export_object_name>[, <export_object_name>[,...] ]
| { ALL | * }
```

```sql
<export_object_name> ::= [<schema_name>.]<identifier> | "*"
```

```sql
<schema_name> ::= <unicode_name>
```
[<schema_name>.]<identifier> specifies objects in a schema for export. Specify <schema_name>."**" to select all objects within the specified schema for export. Specify ALL (without <schema_name>) to select all objects from all schemas in the system for export.

**AS export_format**

Specifies the format to export the data to.

```plaintext
<export_format> ::= (BINARY | CSV)
```

**BINARY**

Table data from column store partitions is exported in internal binary format. Exporting in this format is orders of magnitude faster than exporting the same table in CSV format. When you specify BINARY format for a multistore table, data for dynamic tiering partitions is exported in CSV format with gzip compression.

**CSV**

Table data is exported in CSV format.

**INTO path [ archive_file_name ]**

Specifies the location where export files are placed.

```plaintext
<path> [ <archive_file_name> ] ::= <string_literal>
```

Include the file extension `.tar.gz` or `.tgz` on `<archive_file_name>` to export the data to an archive file.

File locations for extended storage must be on a shared file system available to both SAP HANA and the extended storage system.

For security reasons, the path must not contain symbolic links and must not point inside the database instance folder, except its `backup` and `work` subfolders. Examples of valid values include:

```
'/tmp'
'/hana/shared/HDB/HDB00/backup'
'/hana/shared/HDB/HDB00/work/myexport.tgz'
```

**WITH export_option_list**

A list of export options.

```plaintext
<export_option_list> ::= <export_option>[{, <export_option>}]
```

**<export_option> ::=**

- REPLACE
- CATALOG ONLY
- NO DEPENDENCIES
- THREADS <number_of_threads>
- STATISTICS ONLY
- NO STATISTICS
- TRACE

**REPLACE**

If REPLACE is not specified, then an error is returned if previously exported data exists in the specified export directory.

**CATALOG ONLY**
Specifies to export only the database catalog.

**NO DEPENDENCIES**

Specifies to not export the underlying dependencies of an export object.

**THREADS number_of_threads**

Specifies the number of process threads to be used for concurrent export processing.

\[
\text{<number_of_threads>} \ ::= \text{<unsigned_integer>}
\]

The THREADS parameter specifies how many objects are exported in parallel; the default is 1. Increasing the number of threads reduces export time, but can also negatively affect database system performance. Consider the following items when you use this parameter. When exporting:

- A single table – THREADS has no effect.
- A view or procedure – use two or more threads, up to the number of dependent objects.
- A whole schema – consider using more than 10 threads, with a maximum being the number of CPU cores in the system.
- A whole BW or ERP system database with tens of thousands of tables by using the ALL keyword – you can use up to 256 threads.

**STATISTICS ONLY**

Data statistics objects are exported by default, along with the objects they reference in \(<\text{export_object_name_list}>\) (for example, tables). Export behavior depends on what you specify. If you specify:

- STATISTICS ONLY (no CATALOG ONLY) – both the data and metadata for data statistics objects are exported, as well as the metadata for the objects in \(<\text{export_object_name_list}>\).
- STATISTICS ONLY and CATALOG ONLY – only the metadata for the data statistics objects and the objects in \(<\text{export_object_name_list}>\) are exported; no data is exported.

**NO STATISTICS**

Excludes data statistics objects from the export. Metadata and data for the objects in \(<\text{export_object_name_list}>\) are still exported, and they are impacted as normal by whether CATALOG ONLY is specified.

**TRACE export_trace_file_name**

Writes execution information to the specified trace file when exporting data.

\[
\text{<export_trace_file_name>} \ ::= \text{<string_literal>}
\]

The info level trace file contains detailed error descriptions for troubleshooting. The file is saved to the current trace directory.

The contents of the trace file can be found in the M_TRACEFILE_CONTENTS system view. It contains information on:

- The export list
- Dependencies to the export list
- Feasibility checks on the export
- Metadata exported
- The format of the exported data (binary or CSV)
- The single archive for a single archive export
To delete the trace file, use the ALTER SYSTEM REMOVE TRACES statement.

query_export_specification

Specifies a query to use for the export. See the SAP HANA SQLScript Reference for information on Query Exports.

<query_export_specification> ::=    ON <sqlscript_location_list> FOR <procedure_call_statement>

Permissions

You must have the EXPORT system privilege, and have SELECT privilege on the objects being exported.

Description

The EXPORT command exports tables, views, column views, synonyms, sequences, or procedures in CSV format. Data on the column store can also be exported in BINARY format.

You can only import exported binary files that were generated on an SAP HANA system with the same endianness.

Detailed results of the last execution of the EXPORT statement are stored in the session-local temporary table <current_schema>#EXPORT_RESULT. If no EXPORT has not been executed in the current session, selecting from <current_schema>#EXPORT_RESULT returns an error. If <current_schema> of the session is invalid, then EXPORT throws an exception as it cannot create its result table in <current_schema>.

current_schema.#EXPORT_RESULT

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(255)</td>
<td>Schema of the exported object.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(255)</td>
<td>Name of the exported object.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(16)</td>
<td>Type of the exported object (TABLE, VIEW, and so on).</td>
</tr>
<tr>
<td>LOCATION</td>
<td>VARCHAR(32)</td>
<td>Location (host:port) where the object was exported.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>Export status (done, skipped, failed).</td>
</tr>
<tr>
<td>ERROR</td>
<td>VARCHAR(512)</td>
<td>Error text in case of export failure.</td>
</tr>
</tbody>
</table>

For more information on permissions on the exported files, see SAP HANA Security Checklists and Recommendations.
Examples

Example 1: Export the information from `MY_SCHEMA.T1` in CSV format and replace information previously exported.

```
EXPORT MY_SCHEMA.T1 AS CSV INTO '/tmp' WITH REPLACE;
```

Example 2: Export all the database objects in schema `A` and `B`, replacing any existing export that exists in the `/tmp` directory.

```
EXPORT A."*", B."*" AS CSV INTO '/tmp' WITH REPLACE THREADS 10;
```

Example 3: Export the results of a query where tables `T1` and `T2` are multistore tables.

```
EXPORT ALL AS CSV INTO '/usr/sap/JL01/HDB00/work' FOR SELECT * FROM T1, T2;
```

Example 4: Export an entire directory in compressed format.

```
EXPORT MY_EXTENDED_TABLE AS CSV INTO '/usr/sap/P30/HDB30/work/myexport.tar.gz'
WITH REPLACE;
```

Example 4: Export the schema `my_schema` and create the trace file `export.trc`.

```
EXPORT my_schema."*" INTO '/usr/sap/MS1/HDB01/work' WITH TRACE 'export.trc';
```

Related Information

- SAP HANA Security Checklists and Recommendations
- `M_TRACEFILE_CONTENTS` System View [page 2247]
- `ALTER SYSTEM REMOVE TRACES` Statement (System Management) [page 564]
- `IMPORT` Statement (Multistore Table) [Dynamic Tiering] [page 1414]

5.2.28 GRANT EXTENDED STORAGE ADMIN Statement

[Dynamic Tiering]

Grants the EXTENDED STORAGE ADMIN system privilege required to manage dynamic tiering and create extended storage.

Syntax

```
GRANT EXTENDED STORAGE ADMIN TO <grantee> [WITH ADMIN OPTION]
```
**Syntax Elements**

**WITH ADMIN OPTION**

Specifies that the granted privileges can be granted further by the specified user or by those users having the specified role.

**Permissions**

You must have the EXTENDED STORAGE ADMIN privilege.

**Description**

The EXTENDED STORAGE ADMIN system privilege is required to manage SAP HANA dynamic tiering and create extended storage.

The specified users, roles, objects, and structured privileges have to exist before they can be used in the GRANT command.

To use the GRANT command to grant privileges to other users and roles, a user needs the privilege and also the permissions required to grant that privilege.

Users cannot grant a privilege to themselves.

**System Views**

**USERS**: All users, their creator, creation date and some information about their current status.

**ROLES**: All roles, their creator and creation date.

**GRANTED_ROLES**: Roles that are granted to users or roles.

**GRANTED_PRIVILEGES**: Privileges that are granted to users or roles.

**Examples**

You grant the EXTENDED STORAGE ADMIN privilege to the user worker. You grant these privileges along with the permission for the worker user to grant them further.

```
GRANT EXTENDED STORAGE ADMIN TO worker WITH ADMIN OPTION;
```
5.2.29 IMPORT Statement (Extended Store Table) [Dynamic Tiering]

Imports catalog objects into an extended store table.

Syntax

```
IMPORT <import_object_name_list>     FROM <path> [ [archive_file_name] ]
[ WITH <import_option_list> ]
[ AT <indexserver_host_port> ]
```

Syntax Elements

**import_object_name_list**

Specifies the list of objects to be imported. The ALL and * options are only for use when importing all objects from a specified schema.

```
<import_object_name_list> ::=    <import_object_name>[, <import_object_name>[,...] ]   | { ALL | * }
```

**import_object_name**

Specifies an object to import data from.

```
<import_object_name> ::= [ <schema_name>.]{<identifier> | "*" }
```

```
<schema_name> ::= <identifier>
```

```
[<schema_name>.]<identifier> specifies an object in a schema for import. <schema_name>."*" specifies to select all objects within a schema for import.
```

FROM path [ archive_file_name ]
Specifies the location where the import source is found. Specify `<archive_file_name>` if the import data is in an archive file. This file is uncompressed at runtime as part of the import process. The archive file must have the file extension `.tar.gz` or `.tgz`.

```
<path> [ <archive_file_name> ] ::= <string_literal>
```

File locations for extended storage must be on a shared file system accessible to both SAP HANA and the extended storage system.

When using a distributed system, the FULL_PATH must point to a shared disk. Examples for import path, assuming the database instance is located at:

```
WITH import_option_list
```

Specifies a list of import options.

```
<import_option_list> ::= <import_option>[, <import_option>[,...] ]
```

```
<import_option> ::= REPLACE     | CATALOG ONLY
                  | NO DEPENDENCIES     | THREADS <number_of_threads>
                  | RENAME SCHEMA <rename_schema_list>
                  | FAIL ON INVALID DATA
                  | IGNORE EXISTING
                  | STATISTICS ONLY
                  | NO STATISTICS
                  | TRACE
```

**REPLACE**

Defines the behavior if the import data already exists in the database. When specified, if a table defined in the import data currently exists in the database, then it is dropped and recreated before the data is imported. If the REPLACE option is not specified, then an error is thrown if an existing database table is defined in the import data.

**CATALOG ONLY**

Specifies that only the database catalog should be imported.

This option interacts with the STATISTICS ONLY and NO STATISTICS options.

**DATA ONLY**

Specifies that only the data in the import file should be imported, without updating or changing the metadata. This option is only valid when the specified target object has its own data, such as a table or a data statistics (as opposed to procedures and views, whose data is materialized from other objects at runtime). The target objects must exist in the database and their definition must match that of the data being imported. For CSV data without the REPLACE option, the import data is appended to the existing table data.

When specifying IMPORT `<schema_name>`."*"... and IMPORT ALL... with DATA ONLY, other objects that do not have records are ignored.

This option interacts with the STATISTICS ONLY and NO STATISTICS options.

**NO DEPENDENCIES**

Specifies to not import the underlying dependencies of an import object.

```
THREADS number_of_threads
```
Specifies how many objects are imported in parallel; the default is 1.

<number_of_threads> ::= <unsigned_integer>

Increasing number of threads reduces import time, but can also negatively affect database system performance. Consider the following items when you use this parameter. When importing:

- A single table – THREADS has no effect.
- A view or procedure – use two or more threads, up to the number of dependent objects.
- A whole schema – consider using more than 10 threads, with a maximum being the number of CPU cores in the system.
- A whole BW or ERP system database with tens of thousands of tables by using the ALL keyword – you can use up to 256 threads.

**RENAME SCHEMA rename_schema_list**

Specifies whether to rename the object’s schema during the import.

<rename_schema_list> ::= <rename_schema_token>[(, <rename_schema_token>)]

<rename_schema_token> ::= <source_schema> TO <target_schema>

You cannot specify the same schema as both <source_schema> and <target_schema> in the same or different <rename_schema_token>.

**IGNORE EXISTING**

Does not import objects that already exist in the database. The IGNORE EXISTING option is ignored, if it is used in combination with REPLACE.

**STATISTICS ONLY**

Data statistics objects are imported by default, along with the objects they reference in <import_object_name_list> (for example, tables). When you specify STATISTICS ONLY, the metadata and data for non-statistics objects in <import_object_name_list> are excluded from the import.

If you specify:

- STATISTICS ONLY without DATA ONLY or CATALOG ONLY – then the data and metadata for data statistics objects are imported.
- STATISTICS ONLY and DATA ONLY – then only the data for the data statistics objects is imported.
- STATISTICS ONLY and CATALOG ONLY – then only the metadata for the data statistics objects is imported.

**NO STATISTICS**

Excludes data statistics objects from the import. Metadata and data for non-statistics objects in <import_object_name_list> are still imported, and they are impacted as normal by whether CATALOG ONLY or DATA ONLY is specified.

**TRACE import_trace_file_name**

Writes execution information to the specified trace file when importing data.

TRACE <import_trace_file_name>
<import_trace_file_name> ::= <string_literal>
The info level trace file contains detailed error descriptions for troubleshooting. The file is saved to the current trace directory.

The contents of the trace file can be found in the M_TRACEFILE_CONTENTS system view and contains information on:

- The single archive for the single archive import
- The import list
- Import feasibility checks
- Objects created and dropped during the import
- Schemas created during the import
- Schemas renamed during the import
- Metadata imported for column tables
- The format of the imported data (CSV)
- Indexes, triggers, and foreign keys created during the import
- The results table from the import

To delete the trace file, use the ALTER SYSTEM REMOVE TRACES statement.

indexserver_host_port

Specifies the index server on which tables are created and imported.

<indexserver_host_port> ::= '\<host_name>:<port_number>'

If you specify the hostname and port, the tables are created and imported on there.

If you specify the <host_name> and <port_number>, then the tables are created and the data imported to that location.

Permissions

You must have the IMPORT system privilege and the CREATE ANY privilege on the schema that is being imported into.

Description

The IMPORT statement imports catalog objects (tables, views, synonyms, sequences, and procedures) that have previously been exported with the EXPORT statement (Extended Store Table). To import external data into existing tables use IMPORT statement (Extended Store Table).

The format (BINARY | CSV) of the file being imported is automatically detected.

You can only import exported binary files that were generated on an SAP HANA system with the same endianness.

Detailed results of the last execution of the IMPORT statement are stored in the following session-local temporary table <current_schema>.IMPORT_RESULT. If no IMPORT has not been executed in the current session, selecting from <current_schema>.IMPORT_RESULT returns an error. If <current_schema>
of the session is invalid, then IMPORT throws an exception as it cannot create its result table for the <current_schema>.

current_schema.IMPORT_RESULT

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(255)</td>
<td>The schema of the imported object</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(255)</td>
<td>The name of the imported object</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(16)</td>
<td>The type of the imported object (TABLE, VIEW, and so on)</td>
</tr>
<tr>
<td>LOCATION</td>
<td>VARCHAR(75)</td>
<td>The location (&lt;host&gt;:&lt;port&gt;) where the object was imported</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>The import status (done, skipped, or failed)</td>
</tr>
<tr>
<td>ERROR</td>
<td>VARCHAR(512)</td>
<td>The error text in the case of an import failure</td>
</tr>
</tbody>
</table>

**Example**

Example 1: Import all tables from schema testsch using the REPLACE option and 10 execution threads.

```
IMPORT "testsch".**" AS CSV FROM '/tmp' WITH REPLACE THREADS 10
```

Example 2: Import the schema my_schema and create the trace file import.trc.

```
IMPORT my_schema."**" FROM '/usr/sap/MS1/HDB01/work' WITH TRACE 'import.trc';
```

**Related Information**

ALTERT SYSTEM CANCEL [WORK IN] SESSION Statement (System Management) [page 516]
M_CONNECTIONS System View [page 1785]
SAP Note 2050807
M_TRACEFILE_CONTENTS System View [page 2247]
ALTER SYSTEM REMOVE TRACES Statement (System Management) [page 564]
EXPORT Statement (Extended Store Table) [Dynamic Tiering] [page 1399]
5.2.30 IMPORT Statement (Multistore Table) [Dynamic Tiering]

Imports catalog objects into a multistore table.

Syntax

```
IMPORT <import_object_name_list>        FROM <path> [ <archive_file_name> ]      [ WITH <import_option_list> ]      [ AT <indexserver_host_port>]
```

Syntax Elements

**import_object_name_list**

Specifies the list of objects to be imported. The ALL and * options are only for use when importing all objects from a specified schema.

```
<import_object_name_list> ::=    <import_object_name>, <import_object_name>[,...] ]   | { ALL | * }
```

**import_object_name**

Specifies an object to import data from.

```
<import_object_name> ::= [ <schema_name>,]{<identifier> | *}
```

```
<schema_name> ::= <identifier>
```

```
[<schema_name>,]<identifier> specifies an object in a schema for import. <schema_name>,"*" specifies to select all objects within a schema for import.
```

**FROM path [ archive_file_name ]**

Specifies the location where the import source is found. Specify <archive_file_name> if the import data is in an archive file. This file is uncompressed at runtime as part of the import process. The archive file must have the file extension .tar.gz or .tgz.

```
<path> [ <archive_file_name> ] ::= <string_literal>
```

File locations for extended storage must be on a shared file system accessible to both SAP HANA and the extended store

**WITH import_option_list**

Specifies a list of import options.

```
<import_option_list> ::= <import_option>[, <import_option>[,...] ]
```
REPLACE

Defines the behavior if the import data already exists in the database. When specified, if a table defined in the import data currently exists in the database, then it is dropped and recreated before the data is imported. If the REPLACE option is not specified, then an error is thrown if an existing database table is defined in the import data.

CATALOG ONLY

Specifies that only the database catalog should be imported.

This option interacts with the STATISTICS ONLY and NO STATISTICS options.

DATA ONLY

Specifies that only the data in the import file should be imported, without updating or changing the metadata. This option is only valid when the specified target object has its own data, such as a table or a data statistics (as opposed to procedures and views, whose data is materialized from other objects at runtime). The target objects must exist in the database and their definition must match that of the data being imported. For CSV data without the REPLACE option, the import data is appended to the existing table data.

When specifying IMPORT <schema_name>."*"... and IMPORT ALL... with DATA ONLY, other objects that do not have records are ignored.

This option interacts with the STATISTICS ONLY and NO STATISTICS options.

NO DEPENDENCIES

Specifies to not import the underlying dependencies of an import object.

THREADS number_of_threads

Specifies how many objects are imported in parallel; the default is 1.

Increasing the number of threads reduces import time, but can also negatively affect database system performance. Consider the following items when you use this parameter. When importing:

- A single table – THREADS has no effect.
- A view or procedure – use two or more threads, up to the number of dependent objects.
- A whole schema – consider using more than 10 threads, with a maximum being the number of CPU cores in the system.
- A whole BW or ERP system database with tens of thousands of tables by using the ALL keyword – you can use up to 256 threads.

RENAME SCHEMA rename_schema_list
Specifies whether to rename the object’s schema during the import.

```
<rename_schema_list> ::= <rename_schema_token>[, <rename_schema_token>[,...] ]
```

```
<rename_schema_token> ::= <source_schema> TO <target_schema>
```

You cannot specify the same schema as both `<source_schema>` and `<target_schema>` in the same or different `<rename_schema_token>`.

**FAIL ON INVALID DATA**

Fails the import operation if not all data is imported successfully.

**IGNORE EXISTING**

Does not import objects that already exist in the database. The IGNORE EXISTING option is ignored, if it is used in combination with REPLACE.

**STATISTICS ONLY**

Data statistics objects are imported by default, along with the objects they reference in `<import_object_name_list>` (for example, tables). When you specify STATISTICS ONLY, the metadata and data for non-statistics objects in `<import_object_name_list>` are excluded from the import.

If you specify:

- STATISTICS ONLY without DATA ONLY or CATALOG ONLY – then the data and metadata for data statistics objects are imported
- STATISTICS ONLY and DATA ONLY – then only the data for the data statistics objects is imported
- STATISTICS ONLY and CATALOG ONLY – then only the metadata for the data statistics objects is imported

**NO STATISTICS**

Excludes data statistics objects from the import. Metadata and data for non-statistics objects in `<import_object_name_list>` are still imported, and they are impacted as normal by whether CATALOG ONLY or DATA ONLY is specified.

**TRACE import_trace_file_name**

Writes execution information to the specified trace file when importing data.

```
<import_trace_file_name> ::= <string_literal>
```

The info level trace file contains detailed error descriptions for troubleshooting. The file is saved to the current trace directory.

The contents of the trace file can be found in the M_TRACEFILE_CONTENTS system view and contains information on:

- The single archive for the single archive import
- The import list
- Import feasibility checks
- Objects created and dropped during the import
- Schemas created during the import
- Schemas renamed during the import
The IMPORT statement imports catalog objects (tables, views, synonyms, sequences, and procedures) that have previously been exported with the EXPORT statement (Multistore Table). To import external data into existing tables use IMPORT statement (Multistore Table).

The format (BINARY | CSV) of the file being imported is automatically detected.

You can only import exported binary files that were generated on an SAP HANA system with the same endianness.

Detailed results of the last execution of the IMPORT statement are stored in the following session-local temporary table `<current_schema>#IMPORT_RESULT`. If no IMPORT has not been executed in the current session, selecting from `<current_schema>#IMPORT_RESULT` returns an error. If `<current_schema>` of the session is invalid, then IMPORT throws an exception as it cannot create its result table for the `<current_schema>#IMPORT_RESULT`.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(255)</td>
<td>The schema of the imported object</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(255)</td>
<td>The name of the imported object</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(16)</td>
<td>The type of the imported object (TABLE, VIEW, and so on)</td>
</tr>
<tr>
<td>LOCATION</td>
<td>VARCHAR(75)</td>
<td>The location (<code>&lt;host&gt;:&lt;port&gt;</code>) where the object was imported</td>
</tr>
</tbody>
</table>
### Column Name

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>The import status (done, skipped, or failed)</td>
</tr>
<tr>
<td>ERROR</td>
<td>VARCHAR(512)</td>
<td>The error text in the case of an import failure</td>
</tr>
</tbody>
</table>

Only data from column store partitions of a multistore table can be exported or imported.

### Example

This example imports all tables from schema `testsch` using the REPLACE option and 10 execution threads.

```sql
IMPORT "testsch"."*" AS CSV FROM '/tmp' WITH REPLACE THREADS 10
```

This example imports the schema `my_schema` and creates the trace file `import.trc`.

```sql
IMPORT my_schema."*" FROM '/usr/sap/MS1/HDB01/work' WITH TRACE 'import.trc';
```

### Related Information

- M_TRACEFILE_CONTENTS System View [page 2247]
- ALTER SYSTEM REMOVE TRACES Statement (System Management) [page 564]
- EXPORT Statement (Multistore Table) [Dynamic Tiering] [page 1403]

### 5.2.31 IMPORT FROM Statement (Extended Store Table)

#### [Dynamic Tiering]

Imports external data from a file into an existing extended store table.

### Syntax

```
IMPORT FROM [ <file_type> ] <file_path>
[ INTO <table_name> ]
[ WITH <import_from_option_list> ]
```
Syntax Elements

file_type

The type of the file to be imported.

```
<file_type> ::= CSV FILE | CONTROL FILE
```

You can specify either comma-separated values or control file formats. For more information on CSV and control file formats, see Examples.

file_path

Specifies the complete path and file name of the file to import.

```
<file_path> ::= <string_literal>
```

When `<file_type>` is CSV, a GZIP (.gz) file is supported in `<file_path>` to indicate a GZIP file containing the CSV data.

table_name

Specifies the target table name or data statistics object name into which the imported data will be imported.

```
<database_object_name> ::=    
    [<schema_name>.]<table_name>   
    | STATISTICS [<schema_name>.]<data_statistics_object_name> 
<schema_name> ::= <unicode_name> 
<table_name> ::= <identifier> 
<data_statistics_object_name> ::= <identifier>
```

Specify STATISTICS when importing data into an existing database statistics object. If the data statistics objects already has data, then the data is overwritten by the data being imported. INTO STATISTICS can only be specified in a control file. When importing data to a data statistics object, the only supported `<import_from_option>` option is ERROR LOG.

import_from_option_list

Specifies a list of options that control import behavior.

```
WITH <import_from_option_list>
```

```
<import_from_option_list> ::= <import_from_option> [{, <import_from_option>}...]
```

```
<import_from_option> ::= NO TYPE CHECK     
    | SKIP FIRST <number_of_rows_to_skip> ROW    
    | COLUMN LIST IN FIRST ROW [WITH SCHEMA FLEXIBILITY] 
    | COLUMN LIST ( <column_name_list> ) [WITH SCHEMA FLEXIBILITY] 
    | RECORD DELIMITED BY <string_for_record_delimiter> 
    | FIELD DELIMITED BY <string_for_field_delimiter> 
    | OPTIONALLY ENCLOSED BY <character_for_optional_enclosure> 
    | ESCAPE <escape_character> 
    | DATE FORMAT <string_for_date_format> 
    | TIME FORMAT <string_for_time_format> 
    | TIMESTAMP FORMAT <string_for_timestamp_format> 
    | ERROR LOG <file_path_of_error_log> 
    | FAIL ON INVALID DATA
```

NO TYPE CHECK
Specifies that the record is inserted without checking the type of each field.

**SKIP FIRST number_of_rows_to_skip ROW**

Skips the specified number of rows in the import file.

```sql
SKIP FIRST <number_of_rows_to_skip> ROW
<number_of_rows_to_skip> ::= <unsigned_integer>
```

**COLUMN LIST IN FIRST ROW**

Indicates that the column list is stored in the first row of the CSV import file.

**COLUMN LIST ( column_name_list )**

The column list for the data being imported. The name list has one or more column names. The ordering of the column names has to match the order of the column data in the CSV file and the columns in the target table.

```sql
COLUMN LIST ( <column_name_list> )
<column_name_list> ::= <column_name> [{, <column_name>}...]
<column_name> ::= <identifier>
```

**RECORD DELIMITED BY string_for_record_delimiter**

Specifies the record delimiter used in the CSV file being imported.

```sql
RECORD DELIMITED BY <string_for_record_delimiter>
<string_for_record_delimiter> ::= <string_literal>
```

**FIELD DELIMITED BY string_for_field_delimiter**

Specifies the field delimiter of the CSV file.

```sql
FIELD DELIMITED BY <string_for_field_delimiter>
<string_for_field_delimiter> ::= <string_literal>
```

**OPTIONALLY ENCLOSED BY character_for_optional_enclosure**

Specifies the optional enclosure character that delimits field data.

```sql
OPTIONALLY ENCLOSED BY <character_for_optional_enclosure>
<character_for_optional_enclosure> ::= <character_literal>
```

**ESCAPE escape_character**

Specifies the escape character used in the import data.

```sql
<escape_character> ::= <character_literal>
```

**DATE FORMAT string_for_date_format**

Specifies the format that date strings are encoded with in the import data.

```sql
DATE FORMAT <string_for_date_format>
<string_for_date_format> ::= <string_literal>
```

Values can include:
- Y: year
• MM: month
• MON: name of month
• DD: day

For example:
• 'YYYYMMDD' = 20120520
• 'YYYY-MM-DD' = 2012-05-20
• 'YYYY-MON-DD': 2012-MAY-20

**TIME FORMAT string_for_time_format**

Specifies the format that time strings are encoded with in the import data.

```
TIME FORMAT <string_for_time_format>
<string_for_time_format> ::= <string_literal>
```

Values can include:
• HH24: hour
• MI: minute
• SS: second

For example:
• 'HH24MISS': 143025
• 'HH24:MI:SS': 14:30:25

**TIMESTAMP FORMAT string_for_timestamp_format**

Specifies the format that timestamp strings are encoded with in the import data.

```
TIMESTAMP FORMAT <string_for_timestamp_format>
<string_for_timestamp_format> ::= <string_literal>
```

For example:
• 'YYYY-MM-DD HH24:MI:SS': 2012-05-20 14:30:25

**ERROR LOG file_path_of_error_log**

Creates a HANA error log, DT message log, and DT row log at the specified file path.

```
ERROR LOG <file_path_of_error_log>

<file_path_of_error_log> ::= <string_literal>
```

**FAIL ON INVALID DATA**

When specified, the IMPORT FROM command fails unless all the entries import successfully. If the option is unspecified, entries that contain invalid data are silently ignored.

**Permissions**

You must have the IMPORT privilege.
Description

The IMPORT FROM statement imports external data from a file into an existing table. To import catalog objects (tables, views, etc.) that have been exported with the EXPORT Statement (Extended Store Table) [Dynamic Tiering] [page 1399] statement, use the IMPORT Statement (Extended Store Table) [Dynamic Tiering] [page 1409] statement.

File locations for extended storage must be on a shared file system accessible to both SAP HANA and the extended storage system.

**Note**

All `<string_literal>`s in the `<import_from_option>` support UTF-8 except surrogate-pair encoding.

For security reasons, only CSV files located at paths defined in the `csv_import_path_filter` configuration parameter can be loaded by using the IMPORT FROM SQL statement. This feature can be disabled by using the `enable_csv_import_path_filter` configuration parameter. Two related configuration parameters are specified in the `import_export` section of the `indexserver (nameserver in the case of multi-DB)` configuration, so you can turn off this feature or update path filter as follows:

```sql
ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'system') set ('import_export', 'enable_csv_import_path_filter') = 'false' with reconfigure
ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'system') set ('import_export', 'csv_import_path_filter') = '/A;/B' with reconfigure
```

Once you add a path `/A` to the path filter, every sub-path of `/A` is automatically added as well.

When you perform IMPORT FROM CSV with entries that contain TIME or SECONDDATE values that have fractional second values, dynamic tiering retains the fractions, while SAP HANA does not. This can lead to unexpected failures while selecting data from the tables containing those values. For example, dynamic tiering stores "2017-06-08 21:35:00.109" exactly, where SAP HANA stores "2017-06-08 21:35:00". The two are not equal (the HANA column store value is less than the dynamic tiering extended store value), so that any time a query tries to join those two values — one from a column store partition (or table) and second from an extended store partition (or table) — the result may seem unexpected. See Values and Behaviors of Numeric and Temporal Data Types in the Dynamic Tiering Administration Guide for details.

Examples

Example 1 – Importing data from a single CSV file.

Create a table `mytable1` to store the imported data.

```sql
CREATE COLUMN TABLE mytable1 (A INT, B VARCHAR(10), C DATE, D TIMESTAMP) USING EXTENDED STORAGE;
```

Create a CSV text file `/data/data1.csv` and add the following contents.

```csv
1,"DATA1","2016-05-20","14:30:25"
2,"DATA2","2016-05-21","15:30:25"
3,"DATA3","2016-05-22","16:30:25"
4,"DATA4","2016-05-23","17:30:25"
```
Execute the following to import the data

```sql
IMPORT FROM CSV FILE '/data/data1.csv' INTO mytable1
    WITH RECORD DELIMITED BY '\n'
    FIELD DELIMITED BY ',',;
```

**Example 2** – Importing data from multiple CSV files.

Create a second CSV text file `/data/data2.csv` and add the following contents.

<table>
<thead>
<tr>
<th></th>
<th>DATA1</th>
<th>DATE</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DATA1</td>
<td>2016-05-20</td>
<td>14:30:25</td>
</tr>
<tr>
<td>2</td>
<td>DATA2</td>
<td>2016-05-21</td>
<td>15:30:25</td>
</tr>
<tr>
<td>3</td>
<td>DATA3</td>
<td>2016-05-22</td>
<td>16:30:25</td>
</tr>
<tr>
<td>4</td>
<td>DATA4</td>
<td>2016-05-23</td>
<td>17:30:25</td>
</tr>
</tbody>
</table>

Execute the following to import the data.

```sql
IMPORT FROM CSV FILE '/data/data1.csv', '/data/data2.csv' INTO mytable1
    WITH RECORD DELIMITED BY '\n'
    FIELD DELIMITED BY ',',;
```

**Example 3** – Import using date formats

In the example below, the date format is of the CSV import data is different to the default date format 'YYYY-MM-DD'. In this import data the date format used is 'MM-DD-YYYY'. You create a CSV text file `/data/data_different_date.csv` and add the following contents.

<table>
<thead>
<tr>
<th></th>
<th>DATA1</th>
<th>DATE</th>
<th>TIME</th>
<th>CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DATA1</td>
<td>05-20-2012</td>
<td>14:30:25</td>
<td>123456</td>
</tr>
<tr>
<td>2</td>
<td>DATA2</td>
<td>05-21-2012</td>
<td>15:30:25</td>
<td>234567</td>
</tr>
<tr>
<td>3</td>
<td>DATA3</td>
<td>05-22-2012</td>
<td>16:30:25</td>
<td>345678</td>
</tr>
<tr>
<td>4</td>
<td>DATA4</td>
<td>05-23-2012</td>
<td>17:30:25</td>
<td>456789</td>
</tr>
</tbody>
</table>

You execute the following command to import the data.

```sql
IMPORT FROM CSV FILE '/data/data_different_date.csv' INTO mytable1
    WITH RECORD DELIMITED BY '\n'
    FIELD DELIMITED BY ',',
    DATE FORMAT 'MM-DD-YYYY';
```

**Example 4** – Import using COLUMN LIST

You create a CSV text file `/data/data_col_list.csv` and add the following contents.

<table>
<thead>
<tr>
<th></th>
<th>DATA1</th>
<th>DATE</th>
<th>CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DATA1</td>
<td>2012-05-20</td>
<td>123456</td>
</tr>
<tr>
<td>2</td>
<td>DATA2</td>
<td>2012-05-21</td>
<td>234567</td>
</tr>
<tr>
<td>3</td>
<td>DATA3</td>
<td>2012-05-22</td>
<td>345678</td>
</tr>
<tr>
<td>4</td>
<td>DATA4</td>
<td>2012-05-23</td>
<td>456789</td>
</tr>
</tbody>
</table>

You execute the following commands to import the data using a column list.

```sql
IMPORT FROM CSV FILE '/data/data_col_list.csv' INTO mytable1
    WITH RECORD DELIMITED BY '\n'
    FIELD DELIMITED BY ',',
    COLUMN LIST ('A', 'B', 'C', 'D');
```

**Example 5** – Importing using timestamp format

```sql
IMPORT FROM CSV FILE '/data/data1.csv' INTO mytable1
    WITH RECORD DELIMITED BY '\n'
    FIELD DELIMITED BY ',',
    DATE FORMAT 'MM-DD-YYYY';
```
Example 6 – Importing using a control file

You import the data from Example 1 using a control file you create (/data/data.ctl) that points to the CSV file. The control file contains the following commands:

```
IMPORT DATA INTO TABLE mytable1 FROM '/data/data.csv'
RECORD DELIMITED BY '\n'
FIELD DELIMITED BY ','
OPTIONALLY ENCLOSED BY '"'
ERROR LOG '/data/data.err'
```

You execute the following command to import the data using the control file: `IMPORT FROM CONTROL FILE '/data/data.ctl';`

5.2.32 IMPORT FROM Statement (Multistore Table)  
[Dynamic Tiering]

Imports external data from a file into existing extended storage partitions in a multistore table.

Syntax

```
IMPORT FROM CSV [ <file_type> ] <file_path>
[ INTO <table_name> ]
[ WITH <import_from_option_list> ]
```

Syntax Elements

- **file_type**
  Specifies the type of the file to be imported (comma-separated values or control file format).

  ```
  <file_type> ::= CSV FILE | CONTROL FILE
  ```

  When importing data statistics objects, only import from a control file is supported.

- **file_path**
  Specifies the complete path and file name of the file to import.

  ```
  <file_path> ::= <string_literal>
  ```

  When `<file_type>` is CSV, a GZIP (.gz) file is supported in `<file_path>` to indicate a GZIP file containing the CSV data.

- **table_name**
Specifies the target table name or data statistics object name into which the imported data will be imported.

```
<database_object_name> ::=  
    [<schema_name>.]<table_name>  
    | STATISTICS [<schema_name>.]<data_statistics_object_name>  
<schema_name> ::= <unicode_name>  
<table_name> ::= <identifier>  
<data_statistics_object_name> ::= <identifier>
```

Specify STATISTICS when importing data into an existing database statistics object. If the data statistics objects already has data, then the data is overwritten by the data being imported. INTO STATISTICS can only be specified in a control file. When importing data to a data statistics object, the only supported `<import_from_option>` option is ERROR LOG.

`import_from_option_list`

Specifies a list of options that control import behavior.

```
WITH <import_from_option_list>
<import_from_option_list> ::= <import_from_option>{,, <import_from_option>}...}
```

```
<import_from_option> ::=  
    NO TYPE CHECK  
    | SKIP FIRST <number_of_rows_to_skip> ROW  
    | COLUMN LIST IN FIRST ROW  
    | RECORD DELIMITED BY <string_for_record_delimiter>  
    | FIELD DELIMITED BY <string_for_field_delimiter>  
    | Optionally Enclosed By <character_for_optional_enclosure>  
    | ESCAPE <escape_character>  
    | DATE FORMAT <string_for_date_format>  
    | TIME FORMAT <string_for_time_format>  
    | TIMESTAMP FORMAT <string_for_timestamp_format>  
    | ERROR LOG <file_path_of_error_log>  
    | THREADS <number_of_threads>  
    | FAIL ON INVALID DATA
```

**NO TYPE CHECK**

Specifies that the record is inserted without checking the type of each field.

**SKIP FIRST number_of_rows_to_skip ROW**

Skips the specified number of rows in the import file.

```
SKIP FIRST <number_of_rows_to_skip> ROW  
<number_of_rows_to_skip> ::= <unsigned_integer>
```

**COLUMN LIST IN FIRST ROW**

Indicates that the column list is stored in the first row of the CSV import file.

**WITH SCHEMA FLEXIBILITY** clause is not supported. If specified, an error is returned.

**COLUMN LIST ( column_name_list )**
The column list for the data being imported. The name list has one or more column names. The ordering of the column names has to match the order of the column data in the CSV file and the columns in the target table.

```
COLUMN LIST ( <column_name_list> )

<column_name_list> ::= <column_name> [{, <column_name>}...]

<column_name> ::= <identifier>
```

RECORD DELIMITED BY string_for_record_delimiter

Specifies the record delimiter used in the CSV file being imported.

```
RECORD DELIMITED BY <string_for_record_delimiter>
<string_for_record_delimiter> ::= <string_literal>
```

FIELD DELIMITED BY string_for_field_delimiter

Specifies the field delimiter of the CSV file.

```
FIELD DELIMITED BY <string_for_field_delimiter>
<string_for_field_delimiter> ::= <string_literal>
```

OPTIONALLY ENCLOSED BY character_for_optional_enclosure

Specifies the optional enclosure character that delimits field data.

```
OPTIONALLY ENCLOSED BY <character_for_optional_enclosure>
<character_for_optional_enclosure> ::= <character_literal>
```

ESCAPE escape_character

Specifies the escape character used in the import data.

```
<escape_character> ::= <character_literal>
```

DATE FORMAT string_for_date_format

Specifies the format that date strings are encoded with in the import data.

```
DATE FORMAT <string_for_date_format>
<string_for_date_format> ::= <string_literal>
```

Values can include:
- Y: year
- MM: month
- MON: name of month
- DD: day

For example:
- 'YYYYMMDD' = 20120520
- 'YYYY-MM-DD' = 2012-05-20
- 'YYYY-MON-DD': 2012-MAY-20

TIME FORMAT string_for_time_format
Specifies the format that time strings are encoded with in the import data.

```
TIME FORMAT <string_for_time_format>
<string_for_time_format> ::= <string_literal>
```

Values can include:
- HH24: hour
- MM: minute
- SS: second

For example:
- 'HH24MISS': 143025
- 'HH24:MI:SS': 14:30:25

**TIMESTAMP FORMAT**

```
TIMESTAMP FORMAT <string_for_timestamp_format>
<string_for_timestamp_format> ::= <string_literal>
```

For example:
- 'YYYY-MM-DD HH24:MI:SS': 2012-05-20 14:30:25

**ERROR LOG**

```
ERROR LOG <file_path_of_error_log>
```

Creates a HANA error log, DT message log, and DT row log at the specified file path.

```
<file_path_of_error_log> ::= <string_literal>
```

**THREADS number_of_threads**

Specifies the number of threads that can be used for concurrent import. The default value is 1 and maximum allowed is 256.

**FAIL ON INVALID DATA**

Specifies that the IMPORT FROM command fails unless all the entries import successfully.

### Permissions

You must have the IMPORT privilege.

### Description

The IMPORT FROM statement imports external data from a file into an existing table or data statistics object. To import catalog objects (tables, views, etc.) that have been exported with the EXPORT Statement (Extended Store Table) [Dynamic Tiering] [page 1399] statement, use the IMPORT Statement (Extended Store Table) [Dynamic Tiering] [page 1409] statement.
Restriction

Do not use IMPORT FROM CSV for a multistore table if the table has generated columns, and the CSV file also has data for generated columns.

Optional enclosure, if used, must be a single character.

When importing from data statistics objects, the schema of a data statistics object is the same as that of the schema of the data source on which the data statistics object was created.

File locations for extended storage must be on a shared file system accessible to both SAP HANA and the extended storage system.

i Note

All <string_literal>s in the <import_from_option> support UTF-8 except surrogate-pair encoding.

For security reasons, only CSV files located at paths defined in the csv_import_path_filter configuration parameter can be loaded by using the IMPORT FROM SQL statement. This feature can be disabled by using the enable_csv_import_path_filter configuration parameter. Two related configuration parameters are specified in the import_export section of the indexserver (nameserver in the case of multi-DB) configuration, so you can turn off this feature or update path filter as follows:

```
ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'system') set ('import_export', 'enable_csv_import_path_filter') = 'false' with reconfigure
ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'system') set ('import_export', 'csv_import_path_filter') = '/A;/B' with reconfigure
```

Once you add a path `/A` to the path filter, every sub-path of `/A` is automatically added as well.

If you are importing data into a multistore table that already has data, take a binary export of the table before you import data from new files. If the import fails, you can import the table using the export, and then retry IMPORT FROM CSV after you fix the issue that caused the failure.

When you perform IMPORT FROM CSV with entries that contain TIME or SECONDDATE values that have fractional second values, dynamic tiering retains the fractions, while SAP HANA does not. This can lead to unexpected failures while selecting data from the tables containing those values. For example, dynamic tiering stores "2017-06-08 21:35:00.109" exactly, where SAP HANA stores "2017-06-08 21:35:00". The two are not equal (the HANA column store value is less than the dynamic tiering extended store value), so that any time a query tries to join those two values — one from a column store partition (or table) and second from an extended store partition (or table), such as in a multistore table — the result may seem unexpected. See Values and Behaviors of Numeric and Temporal Data Types in the Dynamic Tiering Administration Guide for details.

Examples

Example 1 – Importing data from a single CSV file.

Create a table mytable1 to store the imported data:

```
CREATE COLUMN TABLE mytable1 (A INT, B VARCHAR(10), C DATE, D TIME)      PARTITION BY RANGE (A) (USING DEFAULT STORAGE (PARTITION 1 <= VALUES < 1000, PARTITION OTHERS)      USING EXTENDED STORAGE (PARTITION 1000 <= VALUES < 10000));
```
Create a CSV text file /data/data1.csv and add the following contents:

1, "DATA1", "2016-05-20", "14:30:25"
2, "DATA2", "2016-05-21", "15:30:25"
3, "DATA3", "2016-05-22", "16:30:25"
4, "DATA4", "2016-05-23", "17:30:25"

Execute the following to import the data:

```
IMPORT FROM CSV FILE '/data/data1.csv' INTO mytable1
    WITH RECORD DELIMITED BY '\n'
    FIELD DELIMITED BY ',',
```

Example 2 – Import using date formats

In the example below, the date format is of the CSV import data is different to the default date format 'YYYY-MM-DD'. In this import data the date format used is 'MM-DD-YYYY'. You create a CSV text file /data/data_different_date.csv and add the following contents:

1, "DATA1", "05-20-2012", "14:30:25", 123456
2, "DATA2", "05-21-2012", "15:30:25", 234567
3, "DATA3", "05-22-2012", "16:30:25", 345678
4, "DATA4", "05-23-2012", "17:30:25", 456789

You execute the following command to import the data:

```
IMPORT FROM CSV FILE '/data/data_different_date.csv' INTO mytable1
    WITH RECORD DELIMITED BY '\n'
    FIELD DELIMITED BY ',',
    DATE FORMAT 'MM-DD-YYYY';
```

Example 3 – Import using COLUMN LIST

You create a CSV text file /data/data_col_list.csv and add the following contents:

1, "DATA1", "2012-05-20", 123456
2, "DATA2", "2012-05-21", 234567
3, "DATA3", "2012-05-22", 345678
4, "DATA4", "2012-05-23", 456789

You execute the following commands to import the data using a column list:

```
IMPORT FROM CSV FILE '/data/data_col_list.csv' INTO mytable1
    WITH RECORD DELIMITED BY '\n'
    FIELD DELIMITED BY ',',
    COLUMN LIST ("A", "B", "C", "D");
```

Example 4 – Importing using timestamp format

```
IMPORT FROM CSV FILE '/data/data1.csv' INTO mytable1
    WITH RECORD DELIMITED BY '\n'
    FIELD DELIMITED BY ',',
    TIMESTAMP FORMAT 'HH:MI:SS';
```

Example 5 – Importing using a control file
You import the data from Example 1 using a control file you create (`/data/data.ctl`) that points to the CSV file. The control file contains the following commands:

```
IMPORT DATA INTO TABLE mytable1 FROM '/data/data.csv'
   RECORD DELIMITED BY '\n'
   FIELD DELIMITED BY ','
   OPTIONALLY ENCLOSED BY '"'
   ERROR LOG '/data/data.err'
```

You execute the following command to import the data using the control file:

```
IMPORT FROM CONTROL FILE '/data/data.ctl';
```

### 5.2.33 INSERT Statement (Extended Store Table) [Dynamic Tiering]

Adds a record to an extended store table.

**Syntax**

```
INSERT INTO <table_name> [ [column_list_clause] ] [ [value_list_clause] | [overriding_clause] ] [subquery] [hint_clause]
```

**Syntax Elements**

- **table_name**
  
  The table where the insert is to be performed, with optional schema name.

  ```
  <table_name> ::= [ <schema_name>. ]<identifier>
  <schema_name> ::= <unicode_name>
  ```

- **column_list_clause**
  
  A list of column identifiers, ordered in the order of values in the `<value_list_clause>` or `<subquery>`.
  
  If the column list is omitted, the database performs the insert using all the columns in the table. A column which is not included in the column list is filled using the columns default value.

  ```
  <column_list_clause> ::= ( <column_name>, ... )
  <column_name> ::= <identifier>
  ```

- **value_list_clause**
  
  A list of values, or expressions evaluating to values, that are inserted into the table.

  ```
  <value_list_clause> ::= VALUES ( <expression>, ... )
  ```

- **subquery**
  
  Additional details for subqueries.
For information on subqueries, see SELECT Statement (Data Manipulation).

**overriding_clause**

This OVERRIDING syntax comes with `<subquery>` and only when the insert table has an identity column. You can force system to use the generated sequence value for identity column by specifying OVERRIDING USER VALUE, and to ignore the value provided from the SELECT. OVERRIDING USER VALUE must be used if the identity column was created with the GENERATED ALWAYS clause and it is optional with GENERATED BY DEFAULT. If OVERRIDING USER VALUE is not specified for a GENERATED BY DEFAULT identity column, the value provided for the column in the SELECT is inserted. OVERRIDING SYSTEM VALUE tells system to use the value from the select for an identity column.

```sql
<overriding_clause> ::= OVERRIDING { SYSTEM | USER } VALUE
```

**hint_clause**

For information on subqueries, see SELECT Statement (Data Manipulation).

**Permissions**

You must have the INSERT object privilege.

**Description**

The values to be inserted can either be values, expressions or the result of a subquery. If the subquery used does not return any records, then the database does not insert any records into the table.

Always define the `<column_list_clause>`. This helps to protect your INSERT queries from damaging data if the target table schema is modified.

**Example**

Create table T:

```sql
CREATE COLUMN TABLE T (KEY INT PRIMARY KEY, VAL1 INT, VAL2 NVARCHAR(20));
```

Insert a row into table T.

```sql
INSERT INTO T VALUES (1, 1, 'The first');
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL1</th>
<th>VAL2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>The first</td>
</tr>
</tbody>
</table>

Insert a new row into table T using column list to specify which columns should receive the input values.

```sql
INSERT INTO T (KEY, VAL2) VALUES (2,3);
```
Insert a row into table T using a subquery.

```
INSERT INTO T SELECT 3, 3, 'The third' FROM DUMMY;
```

Insert values into an array value construction by enumeration.

```
INSERT INTO T1 VALUES ( 1, ARRAY ( 1, 2, 3, 4 ) )
```

Insert values into an array value construction by query.

```
CREATE COLUMN TABLE T0 ( C1 INT )
INSERT INTO T0 VALUES ( 21 )
INSERT INTO T0 VALUES ( 22 )
INSERT INTO T0 VALUES ( 23 )
INSERT INTO T1 VALUES ( 2, ARRAY( SELECT C1 FROM T0 ) )
```

Insert a NULL value into an ARRAY.

```
INSERT INTO T1 (ID) VALUES (3)
```

**Related Information**

SELECT Statement (Data Manipulation) [page 1104]
5.2.34 INSERT Statement (Multistore Table) [Dynamic Tiering]

Syntax

```
INSERT INTO <table_name> [ <column_list_clause> ] { <value_listClause> | <subquery> }
```

Syntax Elements

**table_name**

The table where the insert is to be performed, with optional schema name.

```
<table_name> ::= [ <schema_name>. ]<identifier>
<schema_name> ::= <unicode_name>
```

**column_list_clause**

A list of column identifiers, ordered in the order of values in the `<value_list_clause>` or `<subquery>`.

```
<column_list_clause> ::= ( <column_name>, ... )
<column_name> ::= <identifier>
```

If the column list is omitted, the database performs the insert using all the columns in the table. A column omitted from the column list will be filled using the column default value. For more information on identifiers, see the Introduction in the SAP HANA SQL and System Views Reference.

**value_list_clause**

A list of values, or expressions evaluating to values, that will be inserted into the table.

```
<value_list_clause> ::= VALUES ( <expression>, ... )
```

For descriptions of expressions, see Expressions in the SAP HANA SQL and System Views Reference.

**subquery**

For more information about subqueries, see the SELECT in the SAP HANA SQL and System Views Reference.

**hint_clause**

For more information about hints, see SELECT in the SAP HANA SQL and System Views Reference.
Permissions

You must have the INSERT object privilege.

Description

The INSERT statement adds a record to a multistore table. The values to be inserted can either be values, expressions or the result of a subquery. If the subquery used does not return any records, then the database will not insert any records into the table.

Multistore tables do not support `<hint_clause>`. They do support:

• Single row inserts
• Array inserts using parameters
• INSERT SELECT statements

You should always define the `<column_list_clause>`. This helps to protect your INSERT queries from damaging data if the target table schema is modified.

Example

You create table T:

```sql
CREATE COLUMN TABLE T (VAL1 INT, VAL2 INT, VAL3 DATE, PRIMARY KEY (VAL1, VAL2)) PARTITION BY RANGE (VAL1) ( partition 10 <= values < 20 , partition 20 <=values < 30 ) , RANGE (VAL2) ( USING DEFAULT STORAGE ( PARTITION 10 <=VALUES < 20, PARTITION 20 <= VALUES < 30 ) USING EXTENDED STORAGE ( PARTITION 30 <=VALUES < 50, PARTITION 50 <=VALUES < 100 ))
```

You insert a row into table T.

```sql
INSERT INTO T VALUES (11, 11, '20160101');
```

<table>
<thead>
<tr>
<th>VAL1</th>
<th>VAL2</th>
<th>VAL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>11</td>
<td>Jan 1, 2016</td>
</tr>
</tbody>
</table>

You insert a new row into table T using column list to specify which columns should receive the input values.

```sql
INSERT INTO T (VAL1, VAL2) VALUES (12,13);
```

<table>
<thead>
<tr>
<th>VAL1</th>
<th>VAL2</th>
<th>VAL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>34</td>
<td></td>
</tr>
</tbody>
</table>
You insert a row into table T using a subquery.

```sql
INSERT INTO T SELECT 12, 12, '20160101' FROM DUMMY;
```

<table>
<thead>
<tr>
<th>VAL1</th>
<th>VAL2</th>
<th>VAL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12</td>
<td>January 1, 2016</td>
</tr>
</tbody>
</table>

5.2.35 MERGE DELTA Statement [Dynamic Tiering]

On a delta-enabled extended store or multistore table, MERGE DELTA triggers a merge of data from the delta dbspace to the user dbspace. When used in a multistore table, you can restrict a manual delta merge to just the default store partition or extended store partition of the multistore table.

**Syntax**

```
MERGE DELTA OF <table_name> [PART <n>]     
[WITH PARAMETERS (<parameter_list>, ...)] [FOR {DEFAULT | EXTENDED} STORAGE]
```

**Syntax Elements**

- `table_name`
  The table name where the delta merge will occur, with optional schema name.

  ```
  <table_name> ::= [<schema_name>.]<identifier>
  <schema_name> ::= <unicode_name>
  ```

You can create a multistore table with delta enabled for extended storage partitions and default store partitions.

- `PART n`
  Merge the delta of a specific table partition to the table's main storage.

- `WITH PARAMETERS`
  Passes option that are specific to the column store.

  ```
  <parameter_list> ::= <parameter> [{, <parameter>}...]
  <parameter> ::= <parameter_name> = <parameter_setting>
  ```
<parameter_name> ::= 'SMART_MERGE'

<parameter_setting> ::= 'ON' | 'OFF'

When SMART_MERGE is ON, the database does a smart merge based on merge criteria specified in the
mergedog section of the indexserver configuration.

FOR {DEFAULT | EXTENDED} STORAGE (Optional) Specifies whether to merge just the default store
partitions or the extended store partitions. If you do not specify this parameter, the delta merge is
performed in the default store.

- DEFAULT Restrictions the delta merge to just the default (column) store partition in a multistore table.
- EXTENDED Restrictions the delta merge to just the extended store partition in a multistore table.

Permissions

You must have the UPDATE object privilege on the table.

Description

On a delta-enabled table, MERGE DELTA triggers a merge of data from the delta dbspace to the user dbspace.
As a table is read optimized and compressed, deltas are introduced to optimize insert or update operations. All
insertions are passed to the delta storage. At a certain point in time, delta changes to a table can be merged
into the table main storage.

When used in a multistore table, you can restrict a manual delta merge to just the default store partition or
extended store partition of the multistore table.

The UPDATE privilege on the column store table is required for performing a delta merge.

The behavior of merge types in multistore tables:

<table>
<thead>
<tr>
<th>Merge Type</th>
<th>Behavior for Multistore Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automerge</td>
<td>Automerge is the default setting for delta merges, and is supported in both default store and extended store partitions.</td>
</tr>
<tr>
<td>Smart merge</td>
<td>Specified with the SMART_MERGE parameter, a smart merge determines whether a merge is necessary for the default store. A smart merge is not supported in the extended store partition of a multistore.</td>
</tr>
<tr>
<td>Hard merge</td>
<td>Uses the MERGE DELTA OF '&lt;table_name&gt;' syntax, and performs a merge regardless of store type.</td>
</tr>
</tbody>
</table>
### Merge Type

<table>
<thead>
<tr>
<th>Merge Type</th>
<th>Behavior for Multistore Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forced merge</td>
<td>Specified with the <code>FORCED_MERGE</code> parameter, a forced merge performs a hard merge while disregarding factors like storage size and system resource availability. A forced merge is not supported in the extended store partition of a multistore.</td>
</tr>
<tr>
<td>Memory merge</td>
<td>Specified with the <code>MEMORY_MERGE</code> parameter, performs a delta merge only in memory, and the merge is not persisted. A memory merge is not supported in the extended store partition of a multistore.</td>
</tr>
<tr>
<td>Critical merge</td>
<td>When a database performs a critical merge, it only merges the delta for the default store partition of a multistore table.</td>
</tr>
</tbody>
</table>

### Examples

In these examples, t1 is a multistore table.

The following example merges the delta for the default store partition of the t1 multistore table:

```sql
MERGE DELTA of t1;
```

The behavior is the same if you were to specify the following:

```sql
MERGE DELTA of t1 FOR DEFAULT STORAGE;
```

The following example merges the delta for just the extended store partition of the t1 multistore table:

```sql
MERGE DELTA of t1 FOR EXTENDED STORAGE;
```

The following example triggers a delta merge for partition 32 in table t1. If the partition is an extended store partition, the merge occurs in the extended store:

```sql
MERGE DELTA of t1 PART 32;
```

### 5.2.36 REFRESH STATISTICS Statement (Extended Store Table) [Dynamic Tiering]

Refreshes data statistic objects that the query optimizer uses to make better decisions for query plans.

### Syntax

```sql
REFRESH STATISTICS { <data_statistics_name>[,<data_statistics_name>[,...] ] }
```
ON <data_sources> [ [HAVING] <match_properties> ] };

Syntax Element

data_statistics_name

Specifies the name of the data statistics object.

<data_statistics_name> ::= [schema_name].<identifier>
<schema_name> ::= <identifier>

data_sources

Specifies the data source(s) of the data statistics objects.

<data_sources> ::= <table_name> [ ( <column_name>,<column_name>[,...] ) [ <match_type> ]
<table_name> ::= [schema_name].<identifier>

column_name

Specifies the column for which the data statistics are defined.

<column_name> ::= <identifier>

If no <column_name> is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

match_type

Controls which data statistics objects to match to <data_sources>.

<match_type> ::= EXACT | CASCADE

If <match_type> is not specified, then any data statistics object(s) that reference all or some of the columns, but no other columns specified in <data_sources> are refreshed.

Specify EXACT to refresh data statistics objects that precisely match <data_sources> (including column order).

Specify CASCADE to refresh data statistics objects that reference at least one column in <data_sources>. 
Use this table to understand how matching is performed based on `<match_type>` when `<data_sources>` is T(A, B, C):

<table>
<thead>
<tr>
<th>Match type</th>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>(not specified)</td>
<td>T(A,C) T(C) T(B,A)</td>
<td>T(A,X) - because T(X) is not a column in <code>&lt;data_sources&gt;</code>.</td>
</tr>
<tr>
<td>EXACT</td>
<td>T(A,B,C)</td>
<td>T(B,A,C) - because the column order is different than the column order of <code>&lt;data_sources&gt;</code>. T(A) - because it does not contain the exact same columns and column order of <code>&lt;data_sources&gt;</code>. T(X,A,B,C) - because T(X) is not a column in <code>&lt;data_sources&gt;</code>.</td>
</tr>
<tr>
<td>CASCADE</td>
<td>T(A,C) T(C) T(B,A) T(A,B,C) T(A,X) T(B,C) T(A) T(C,B,A,X)</td>
<td>T(X) - because it does not contain any columns that match the columns in <code>&lt;data_sources&gt;</code>.</td>
</tr>
</tbody>
</table>

**match_properties**

Specifies properties to use for matching when selecting data statistics.

```sql
<match_properties> ::= <match_property>[...]  
<match_property> ::=  
  TYPE <data_statistics_type>  | REFRESH TYPE <refresh_type_filter>
```

If TYPE is not specified, then all data statistics objects of any type on the specified data sources are refreshed (ALL). For descriptions of the supported data statistics types see the `CREATE STATISTICS Statement (Extended Store Table)` topic.

**data_statistics_type**

Specifies the type of data statistics objects to match when selecting the data statistics.

```sql
<data_statistics_type> ::= TYPE <type_name>  
[type_name] ::=  
  HISTOGRAM  
  SIMPLE  
  TOPK  
  SKETCH
```
refresh_type_filter

Specifies the refresh strategy to match in the data statistics objects when selecting data statistics. ALL is the default.

<refresh_type_filter> ::= AUTO | MANUAL | ALL

Permissions

One of the following is true:

- You own the object you are refreshing statistics on.
- You have the ALTER privilege on the object you are refreshing statistics on.

Description

The REFRESH STATISTICS statement rebuilds the specified data statistics objects. REFRESH STATISTICS can be executed for a list of named data statistics objects, or for objects that approximate the specified <data_sources>.

The REFRESH STATISTICS statement only affects enabled data statistics objects.

See the CREATE STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1361] Statement for descriptions of the supported data statistic types (<data_statistics_type>).

For more information on schema names and identifiers, see Identifiers [page 28].

Example

The following example refreshes the HISTOGRAM on column T(X), where T is a virtual table:

    REFRESH STATISTICS ON MYSHEMA.T(X) TYPE HISTOGRAM;

The following example refreshes any SKETCH with <data_sources> containing one or more of columns T(X), T(Y), or T(Z) (only) in any order

    REFRESH STATISTICS ON MYSHEMA.T(X, Y, Z) TYPE SKETCH;

The following example refreshes the database statistics object named HISTOGRAM_T_X:

    REFRESH STATISTICS MYSHEMA.HISTOGRAM_T_X;
The following example refreshes any data statistic object with `<data_sources>` defined as T(A, B). HISTOGRAMS on T(A) and on T(B) would not be affected.

```sql
REFRESH STATISTICS ON MYSHEMA.T(A, B) EXACT;
```

The following example refreshes any data statistic object that has table columns T(A), T(B), or T(C) listed in `<data_sources>`:

```sql
REFRESH STATISTICS ON MYSHEMA.T(A, B, C) CASCADE TYPE ALL;
```

The following example manually refreshes the HISTOGRAM MYSHEMA.MYHISTOGRAM on table column MYSHEMA.T(A):

```sql
REFRESH STATISTICS MYSHEMA.MYHISTOGRAM ON MYSHEMA.T(A) TYPE HISTOGRAM REFRESH TYPE MANUAL;
```

The following example refreshes the row count on table T:

```sql
REFRESH STATISTICS ON T TYPE RECORD COUNT;
```

**Related Information**

- CREATE STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1361]
- ALTER STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1299]
- DROP STATISTICS Statement (Extended Store Table) [Dynamic Tiering] [page 1393]
- M_DATA_STATISTICS System View [page 1851]
- M_SYSTEM_DATA_STATISTICS System View [page 2189]

**5.2.37 REFRESH STATISTICS Statement (Multistore Table) [Dynamic Tiering]**

Refreshes data statistic objects that the query optimizer uses to make better decisions for query plans.

**Syntax**

```sql
REFRESH STATISTICS { <data_statistics_name>[,<data_statistics_name>[,...] ] | ON <data_sources> [ [HAVING] <match_properties> ] };
```

**Syntax Element**

- `data_statistics_name`
Specifies the name of the data statistics object.

```sql
<data_statistics_name> ::= [<schema_name>].<identifier>
<schema_name> ::= <identifier>
```

data_sources

Specifies the data source(s) of the data statistics objects.

```sql
<data_sources> ::=<table_name> [ ( <column_name>[, <column_name>[,...] ] ) ]
[ FOR { DEFAULT | EXTENDED } STORAGE ] [ <match_type> ]
```

table_name

Specifies the table on which the data statistics are defined.

```sql
<table_name> ::= [<schema_name>].<identifier>
```

column_name

Specifies the column for which the data statistics are defined.

```sql
<column_name> ::= <identifier>
```

If no `<column_name>` is specified, then all statistics for the table that match the specified properties are altered, including table-wide statistics (RECORD COUNT).

**FOR {DEFAULT | EXTENDED} STORAGE**

The STORAGE clause can be specified only for multistore tables. Specify DEFAULT when the data statistics object is created on the column store part. Specify EXTENDED STORAGE when the data statistics object is created on the extended storage part. If you do not specify anything, then two data statistics objects are created, one on the extended storage part and one on the column storage part.

Statistics of some types can only be created on one storage class partition; for example, SKETCH on HANA column store partitions, and RECORD COUNT on extended storage partitions. Attempting to create such objects on a multistore table without specifying storage type returns an error.

You can implement different statistics types for different storage classes, and create statistics to address specific performance problems that only affect the query plan for one of the storage classes of the table. The statement returns an error if no partitions of the specified storage type exist.

```
c
```

**match_type**

Controls which data statistics objects to match to `<data_sources>`.

```sql
<match_type> ::= EXACT | CASCADE
```

If `<match_type>` is not specified, then any data statistics object(s) that reference all or some of the columns, but no other columns specified in `<data_sources>` are refreshed.

Specify EXACT to refresh data statistics objects that precisely match `<data_sources>` (including column order).

Specify CASCADE to refresh data statistics objects that reference at least one column in `<data_sources>`.
Use this table to understand how matching is performed based on `<match_type>` when `<data_sources>` is T(A, B, C):

<table>
<thead>
<tr>
<th>Match type</th>
<th>Example matches</th>
<th>Example non-matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>(not specified)</td>
<td>T(A,C)</td>
<td>T(A,X) - because T(X) is not a column in <code>&lt;data_sources&gt;</code>.</td>
</tr>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,A)</td>
<td></td>
</tr>
<tr>
<td>EXACT</td>
<td>T(A,B,C)</td>
<td>T(B,A,C) - because the column order is different than the column order of <code>&lt;data_sources&gt;</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T(A) - because it does not contain the exact same columns and column order of <code>&lt;data_sources&gt;</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T(X) - because it does not contain any columns that match the columns in <code>&lt;data_sources&gt;</code>.</td>
</tr>
<tr>
<td>CASCADE</td>
<td>T(A,C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A,B)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A,B,C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A,X)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(B,C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(A)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T(C,B,A,X)</td>
<td></td>
</tr>
</tbody>
</table>

**match_properties**

Specifies properties to use for matching when selecting data statistics.

```
<match_properties> ::= <match_property>[...]
<match_property> ::= TYPE <data_statistics_type> | REFRESH TYPE <refresh_type_filter>
```

If TYPE is not specified, then all data statistics objects of any type on the specified data sources are altered (ALL). For descriptions of the supported data statistics types see the CREATE STATISTICS Statement (Multistore Table) topic.

**data_statistics_type**

Specifies the type of data statistics objects to match when selecting the data statistics.

```
<data_statistics_type> ::= TYPE <type_name>
<type_name> ::= TYPE HISTOGRAM | SIMPLE | TOPK | SKETCH
```
refresh_type_filter

Specifies the refresh strategy to match in the data statistics objects when selecting data statistics. ALL is the default.

\[
<\text{refresh\_type\_filter}> ::= \text{AUTO} \mid \text{MANUAL} \mid \text{ALL}
\]

Permissions

One of the following is true:

• You own the object you are refreshing statistics on.
• You have the ALTER privilege on the object you are refreshing statistics on.

Description

The REFRESH STATISTICS statement rebuilds data statistics objects that approximate the specified \(<\text{data\_sources}>\).

The REFRESH STATISTICS statement only affects enabled data statistics objects.

See the CREATE STATISTICS [page 815] Statement for descriptions of the supported data statistic types \(<\text{data\_statistics\_type}>\).

For more information on schema names and identifiers, see Identifiers [page 28].

Example

The following example refreshes the HISTOGRAM on column T(X), where T is a virtual table:

\[
\text{REFRESH STATISTICS ON MYSCHEMA.T(X) TYPE HISTOGRAM;}
\]

The following example refreshes any SKETCH with \(<\text{data\_sources}>\) containing one or more of columns T(X), T(Y), or T(Z) (only) in any order

\[
\text{REFRESH STATISTICS ON MYSCHEMA.T(X, Y, Z) TYPE SKETCH;}
\]

The following example refreshes the database statistics object named HISTOGRAM_T_X:

\[
\text{REFRESH STATISTICS MYSCHEMA.HISTOGRAM_T_X;}
\]
The following example refreshes any data statistic object with `<data_sources>` defined as T(A, B). HISTOGRAMS on T(A) and on T(B) would not be affected.

```sql
REFRESH STATISTICS ON MYSHEMA.T(A, B) EXACT;
```

The following example refreshes any data statistic object that has table columns T(A), T(B), or T(C) listed in `<data_sources>`:

```sql
REFRESH STATISTICS ON MYSHEMA.T(A, B, C) CASCADE TYPE ALL;
```

The following example manually refreshes the HISTOGRAM MYSHEMA.MYHISTOGRAM on table column MYSHEMA.T(a):

```sql
REFRESH STATISTICS MYSHEMA.MYHISTOGRAM ON MYSHEMA.T(a) TYPE HISTOGRAM REFRESH TYPE MANUAL;
```

The following example refreshes the row count on table T:

```sql
REFRESH STATISTICS ON T TYPE RECORD COUNT;
```

**Related Information**

CREATE STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1366]
ALTER STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1305]
DROP STATISTICS Statement (Multistore Table) [Dynamic Tiering] [page 1395]
M_DATA_STATISTICS System View [page 1851]
M_SYSTEM_DATA_STATISTICS System View [page 2189]

### 5.2.38 UPDATE Statement (Extended Store Table) [Dynamic Tiering]

Changes the values of the records of an extended store table.

**Syntax**

```sql
UPDATE [<top_clause>] <table_name> [<alias_name>]
   <set_clause> [ WHERE <condition> ] [<hint_clause>]
```

**Syntax Elements**

- `table_name`
The table name where the UPDATE is to be performed, with optional schema name.

\[
\text{<table_name>} ::= [\text{<schema_name>}.]\text{<identifier>}
\]

\[
\text{<schema_name>} ::= \text{<unicode_name>}
\]

**alias_name**

An alias that can be used to refer to the table defined by `<table_name>`.

\[
\text{<alias_name>} ::= [\text{AS}] \text{<identifier>}
\]

For more information about identifiers, see *Introduction in Introduction to SQL* [page 27].

**set_clause**

The column names and associated values that are to be set by the update statement.

\[
\text{<set_clause>} ::= \text{SET} \{ \text{<column_name> = <expression>} \mid (\text{<with_clause> <subquery>} ) \},...
\]

For descriptions of expressions, see the *Expressions* [page 56] section in this guide. For the definitions of `<with_clause>` and `<subquery>`, see *SELECT Statement (Data Manipulation)* [page 1104]. The `<with_clause>` can be used with column names, not with column lists.

**condition**

Specifies the conditions where the command should be performed.

\[
\text{WHERE} \text{ <condition>} \]

\[
\text{<condition>} ::= \text{ <condition>} \text{ OR} \text{ <condition>} \mid \text{ <condition>} \text{ AND} \text{ <condition>} \mid \text{ NOT} \text{ <condition>} \mid (\text{ <condition>} ) \mid \text{ <predicate>}
\]

For more information about predicates, see the *Predicates* [page 61] section in this guide.

**hint_clause**

For more information about hints, see *SELECT Statement (Data Manipulation)* [page 1104] in this guide.

**top_clause**

Allows you to limit the number of updated records, for example for chunkwise updates.

\[
\text{<top_clause>} ::= \text{TOP} \text{ <unsigned_integer>}
\]

\[
\text{<unsigned_integer>}\text{ defines the number of records updated in one chunk. This parameter is not supported for variants of UPDATE statements like UPDATE MERGE DELTA, UPDATE PRELOAD, UPDATE UNLOAD and UPDATE MOVE.}
\]

**Permissions**

You must have the UPDATE object privilege.
**Description**

The UPDATE statement changes the values of the records of an extended store table.

If the WHERE condition is used and is true for a specific row, the update is performed on that row. If the WHERE clause is omitted, the UPDATE command updates all records of a table.

**Examples**

You create table T, and insert two rows into it.

```
CREATE COLUMN TABLE T (KEY INT PRIMARY KEY, VAL INT);
INSERT INTO T VALUES (1, 1);
INSERT INTO T VALUES (2, 2);
```

You update the rows of table T if the condition in the WHERE clause is true.

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

You update all rows of table T because a where clause is not specified as part of the update statement.

```
UPDATE T SET VAL = KEY + 10;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
</tbody>
</table>

You create table T2, and insert two rows into it.

```
CREATE COLUMN TABLE T2 (KEY INT PRIMARY KEY, VAR INT);
INSERT INTO T2 VALUES (1, 2);
INSERT INTO T2 VALUES (3, 6);
```

You update the values of table T by joining the target table T with table T2.

```
UPDATE T SET VAL = T2.VAR FROM T, T2 WHERE T.KEY = T2.KEY;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
You update a chunk of 100,000 records of the table `testtab` that contain 'XXX' in the `request` column and the value 0 in the `updated` column. For each row processed, the `updated` column is set to 1.

```sql
update top 100000 testtab set updated = 1 where request = 'XXX' and updated = 0
```

You UPDATE a table with an array value construction by enumeration.

```sql
UPDATE T1 SET C1 = ARRAY ( 11, 12, 13, 14 ) WHERE ID = 1
```

You UPDATE a table with an array value construction by query.

```sql
UPDATE T1 SET C1 = ARRAY( SELECT C1 FROM T0 ) WHERE ID = 1
```

### 5.2.39 UPDATE Statement (Multistore Table) [Dynamic Tiering]

Changes the values of the records of a multistore table.

#### Syntax

```sql
UPDATE [<top_clause>] <table_name> [<alias_name>] [<partition_restriction>] <set_clause> [ WHERE <condition> ] [<hint_clause>]
```

#### Syntax Elements

- **table_name**
  
  The table name where the UPDATE is to be performed, with optional schema name.

  ```sql
  <table_name> ::= [<schema_name>.]<identifier>
  <schema_name> ::= <unicode_name>
  ```

- **alias_name**
  
  An alias that can be used to refer to the table defined by `table_name`.

  ```sql
  <alias_name> ::= [AS] <identifier>
  ```

  For more information about identifiers, see `Introduction` in *Introduction to SQL* [page 27]

- **partition_restriction**
  
  For the default (column store) partitions of multistore tables, `<partition_restriction>` specifies the partition in which the target updated rows are located.

  ```sql
  <partition_restriction> ::= PARTITION ( <partition_number> )
  <partition_number> ::= <unsigned_integer>
  ```
If you do not specify a partition restriction, then the database checks all partitions for the update. But if there is partition restriction, only the rows in specified partition are updated.

Functional restrictions:

• An UPDATE statement with a partition restriction is blocked if the following are true:
  1. Partitioning key column update
  2. UNIQUE constraint column update
  3. PRIMARY KEY column update
• Updates that change the updated rows partition are not allowed.
• <partition_restriction> is not supported if the partition number in the list contains a partition ID of the extended store.

set_clause

The column names and associated values that are to be set by the update statement.

```
<set_clause> ::= SET {<column_name> = <expression> | {<with_clause><subquery>}},...
```

For descriptions of expressions, see Expressions [page 56]. For the definitions of <with_clause> and <subquery>, see SELECT Statement (Data Manipulation) [page 1104]. The <with_clause> can be used with column names, not with column lists.

condition

Specifies the conditions where the command should be performed.

```
WHERE <condition>
<condition> ::= <condition> OR <condition>
  | <condition> AND <condition>
  | NOT <condition>
  | ( <condition> )
  | <predicate>
```

For more information about predicates, see Predicates [page 61].

For multistore tables, UPDATE supports only simple set expressions, and disallows complex expressions. UPDATE on multistore tables allows only simple predicates in the WHERE clause, and disallows subqueries in the WHERE clause.

hint_clause

For more information about hints, see SELECT Statement (Data Manipulation) [page 1104].

top_clause

Limits the number of updated records.

```
<top_clause> ::= TOP <unsigned_integer>
```

For example, for chunkwise updates, <unsigned_integer> defines the number of records updated in one chunk. Variants of UPDATE statements like UPDATE MERGE DELTA, UPDATE PRELOAD, UPDATE UNLOAD and UPDATE MOVE do not support this parameter.

Permissions

You must have the UPDATE object privilege.
Description

The UPDATE statement changes the values of the records of a table.

If the WHERE condition is used and is true for a specific row, the update is performed on that row. If the WHERE clause is omitted, the UPDATE command updates all records of a table.

You can update non-partitioning columns in a multistore table. You can update aging columns, partition keys, and primary key columns without restriction.

You can use UPDATE to move data from the catalog store to the extended store and back. Any update statement for a column store table works for a multistore table.

Examples

Create table T, and insert two rows into it.

```sql
CREATE COLUMN TABLE T (KEY INT PRIMARY KEY, VAL INT)  PARTITION BY RANGE (KEY) ( USING DEFAULT STORAGE
(PARTITION '0' <= VALUES < '40') USING EXTENDED STORAGE
(PARTITION '40' <= VALUES < '80') )
INSERT INTO T VALUES (1, 1);  INSERT INTO T VALUES (2, 2);
```

Update the rows of table T if the condition in the WHERE clause is true.

```sql
UPDATE T SET VAL = VAL + 1 WHERE KEY = 1;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Update all rows of table T because a where clause is not specified as part of the update statement.

```sql
UPDATE T SET VAL = KEY + 10;
```

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
</tbody>
</table>

Create table T2, and insert two rows into it.

```sql
CREATE COLUMN TABLE T2 (KEY INT PRIMARY KEY, VAL INT)  PARTITION BY RANGE (KEY) ( USING DEFAULT STORAGE
(PARTITION '0' <= VALUES < '40') USING EXTENDED STORAGE
(PARTITION '40' <= VALUES < '80') )
INSERT INTO T2 VALUES (1, 2);
```
**INSERT INTO T2 VALUES (3, 6);**

Update the values of table T by joining the target table T with table T2.

**UPDATE T SET VAL = T2.VAR FROM T, T2 WHERE T.KEY = T2.KEY;**

<table>
<thead>
<tr>
<th>KEY</th>
<th>VAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Update a chunk of 100,000 records of the table testtab that contain 'XXX' in the request column and the value 0 in the updated column. For each row processed, the updated column is set to 1.

**update top 100000 testtab set updated = 1 where request = 'XXX' and updated = 0**

Update an aging column.

**update table t1 set c = 10 where t1.temperature = '0000000'**

### 5.2.40 Additional Syntax [Dynamic Tiering]

SQL syntax that matches SAP HANA core syntax.

**Related Information**

- ALTER SYSTEM ALTER TABLE PLACEMENT Statement (System Management) [page 508]
- ALTER SYSTEM CANCEL [WORK IN] SESSION Statement (System Management) [page 516]
- ALTER SYSTEM STOP SERVICE Statement (System Management) [page 585]
- CREATE REMOTE SOURCE Statement (Access Control) [page 799]
- CREATE SCHEMA Statement (Data Definition) [page 807]
- CREATE TRIGGER Statement (Data Definition) [page 876]
- DELETE Statement (Data Manipulation) [page 938]
- DROP AUDIT POLICY Statement (Access Control) [page 946]
- DROP SCHEMA Statement (Data Definition) [page 971]
- DROP TABLE Statement (Data Definition) [page 980]
- DROP TRIGGER Statement (Data Definition) [page 982]
- EXPLAIN PLAN Statement (Data Manipulation) [page 992]
- RENAME TABLE Statement (Data Definition) [page 1094]
- REPLACE Statement (Data Manipulation) [page 1096]
- REVOKE Statement (Access Control) [page 1098]
- SELECT Statement (Data Manipulation) [page 1104]
5.3 Streaming Analytics

⚠️ Caution
This guide contains syntax variations for different product contexts. These are handled as separate reference topics that have titles that announce the context in square brackets at the end of the title. The guide is also organized to keep reference topics together for a context. Always be sure you are using the SQL reference topic specific to your context.

5.3.1 ALTER DATABASE Statement [Streaming Analytics]

Alter an existing database to add or remove the streamingserver service.

**Syntax**

```
ALTER DATABASE <database_name>      
  ADD 'streamingserver' [ AT [LOCATION] '<hostname>[:<port>]']  
  | REMOVE 'streamingserver' AT [LOCATION] '<hostname>[:<port>]'
```

**Syntax Elements**

- `database_name` Specifies the name of the database to which the change is applied.

  - `<database_name> ::= <identifier>`

- `hostname` Specifies the name of the dedicated streaming analytics host.

- `port` Specifies the port number where the streaming analytics service runs. Port is comprised of 3xx16, where xx is the instance number.

  - `<hostname> ::= <identifier>`

**Permissions**

You must have the DATABASE ADMIN privilege.
Description

This statement alters an existing database by adding or removing the `streamingserver` service.

Examples

The following code example shows you how to add the service `streamingserver` to the database `my_database` at hostA, for instance 03.

```
ALTER DATABASE my_database ADD 'streamingserver' AT LOCATION 'hostA:30316'
```

The following code example shows you how to remove the service `streamingserver` from the database `my_database` at hostA, for instance 03..

```
ALTER DATABASE my_database REMOVE 'streamingserver' AT LOCATION 'hostA:30316'
```

5.3.2 ALTER SYSTEM INITIALIZE SERVICE Statement
[Streaming Analytics]

Creates the remote sources and users, and imports the delivery unit (DU) for the service.

Syntax

```
ALTER SYSTEM INITIALIZE SERVICE 'streamingserver'
   [ AT [ LOCATION ] '<hostname>[:<port>]'
   [ WITH CREDENTIAL TYPE 'PASSWORD' USING <streaming_cluster_password> ] ]
```

Syntax Elements

- `hostname` Specifies the name of the dedicated streaming analytics host.
- `port` Specifies the port number where the streaming analytics service runs. Port is comprised of 3xx16, where xx is the instance number.
- `streaming_cluster_password` Specifies the password used to manage the streaming analytics cluster.
Permissions

You must have the SERVICE ADMIN privilege.

Description

This statement creates the remote sources and users, and imports the delivery unit (DU) for the service.

Example

This example initialized the `streamingserver` service on host `mysdshostname`, port `30016`, with the streaming cluster password = MySecretPass. 00 in the port number is the instance number of the system.

```
ALTER SYSTEM INITIALIZE SERVICE 'streamingserver' at 'mysdshostname:30016'
with CREDENTIAL TYPE 'PASSWORD' USING 'MyStreamingClusterPassword'
```

5.3.3 ALTER SYSTEM UNINITIALIZE SERVICE Statement

[Streaming Analytics]

Removes the remote sources and users, and removes the delivery units (DU) for the `streamingserver` service.

Syntax

```
ALTER SYSTEM UNINITIALIZE SERVICE 'streamingserver'
  [ AT [ LOCATION] <hostname>:<port> ]
  [ DROP <drop_option> ]
```

Syntax Elements

- **hostname** Specifies the name of the dedicated streaming analytics host.
- **port** Specifies the port number where the streaming analytics service runs. Port is comprised of 3xx16, where xx is the instance number.
- **drop_option**
Drops the streaming server service.

\[
\text{<drop_option>} ::= \text{CASCADE} | \text{RESTRICT}
\]

**CASCADE**
Drops the streaming service and any dependent streaming projects.

**RESTRICT**
Drops the streaming service only when dependent streaming projects do not exist. If this drop option is used and a dependent streaming projects exists an error is returned and the service is not uninitialized.

**Permissions**

You must have the SERVICE ADMIN privilege.

**Description**

This statement drops the specified streaming server and its dependent data, technical users, projects, and removes the delivery units (DU) for the service.

**Examples**

The following code example drops a streaming server and its technical users and data even if dependent streaming projects exist.

\[
\text{ALTER SYSTEM UNINITIALIZE SERVICE 'streamingserver' drop cascade;}
\]

### 5.4 Text Mining

⚠️ **Caution**

This guide contains syntax variations for different product contexts. These are handled as separate reference topics that have titles that announce the context in square brackets at the end of the title. The guide is also organized to keep reference topics together for a context. Always be sure you are using the SQL reference topic specific to your context.

- **Text Mining: TM_CATEGORIZE_KNN Function [page 1456]**
  
  This function classifies (categorizes) an input document with respect to sets of categories (taxonomies).

- **Text Mining: TM_GETRELATEDDOCUMENTS Function [page 1460]**
This text mining function returns the top-ranked related documents for a query document within a search request and stores these documents (including metadata) in the return table.

Text Mining: TM_GETRELATED_TERMS Function [page 1463]
This text mining function returns the top-ranked related terms for a query term, based on a set of reference documents.

Text Mining: TM_GETRELEVANTDOCUMENTS Function [page 1467]
This text mining function returns the top-ranked documents that are relevant to a term.

Text Mining: TM_GETRELEVANTTERMS Function [page 1470]
This text mining function returns the top-ranked relevant terms that describe a document.

Text Mining: TM_GETSUGGESTEDTERMS Function [page 1473]
This text mining function returns the top-ranked terms that match an initial substring. This function can be used for type-ahead or auto-completion functions.

Related Information

SAP HANA Text Mining Developer Guide

5.4.1 Text Mining: TM_CATEGORIZE_KNN Function

This function classifies (categorizes) an input document with respect to sets of categories (taxonomies).

⚠️ Caution
The information in this topic applies only to the SAP HANA context cited in the title of the topic.

In the following texts, the terms category, categorization and taxonomy are used instead of class, classification, and classification system. The latter terms may be considered synonyms of the former terms.

Syntax

```
TM_CATEGORIZE_KNN (      <tm_document>       <tm_search_categorize_knn>      { <tm_return_category>, ... }  )
```

Specifies the document you want to categorize, the reference documents, and the taxonomies (category schemata) to be used for categorization.
Syntax Elements

The function uses the following syntax elements: `<tm_document>`, `<tm_search_categorize_knn>`, `<tm_return_category>`. 

```sql
<tm_document> ::= DOCUMENT { <string> [ LANGUAGE <string> ] [ MIME TYPE <string> ]
| ( <subquery> ) [ LANGUAGE <string> ] [ MIME TYPE <string> ]
| IN FULLTEXT INDEX WHERE <condition> }
```

The document to be categorized. There are three ways to specify the document:

1. Specify the document text as a string. LANGUAGE and MIME TYPE can be specified to indicate how the document is processed. If nothing is specified, then the default language (English) and mime type (TEXT/PLAIN) will be used.

2. Specify a SELECT query that returns the document text. LANGUAGE and MIME TYPE can be specified as optional parameters.

3. If the query document is part of the full-text index, which was constructed for the set of reference documents, then it can be referred to by a WHERE condition. This is much more efficient since the processed document is used immediately.

```sql
<tm_search_categorize_knn> ::= SEARCH NEAREST NEIGHBORS { <knn_int> | DEFAULT } <reference_column>
FROM <reference_table>
[ WHERE <condition> ]
[ WITH TERM TYPE <string>, … ]
```

Indicates the number of nearest neighbors for KNN categorization in `<knn_int>`. DEFAULT means to use the value set at text mining initialization. Specifies the set of reference documents in `<reference_column>` and `<reference_table>`. The specified column must be of type text or must have a full-text index. The set of reference documents can be restricted by WHERE `<condition>`, which means that only those documents will be used in the calculations. The set of reference terms can be restricted by WITH TERM TYPE, which means that only those terms will be used in the calculations. For example WITH TERM TYPE 'proper*','noun' only considers terms that are proper names or nouns. The termTypeRestriction property in the text mining configuration determines which term types are available.

The best value for the parameter `<knn_int>` must be determined empirically for each document set. Setting this parameter to a value in the range 5 – 25 will usually yield good results. For categories with only a few documents, a lower `<knn_int>` may yield better results.

```sql
<tm_return_category> ::= RETURN TOP { <top_int> | DEFAULT } <category_column>
FROM <category_table>
[ JOIN <reference_column> ON <pk_of_category_table> = <pk_of_reference_table> ]
```

The parameter `<top_int>` specifies the maximum number of category results to return for the category column. DEFAULT means to use the value set at text mining initialization. `<category_column>` contains the category labels. It may be a column in the same table as the reference documents or a column in a different table. In the second case, a join clause must be specified, in order to indicate how the category label(s) for particular documents are determined.
For example:

```
RETURN TOP 3 "topics" FROM "Documents"."NewsTopics" JOIN "Documents"."News" ON "Documents"."News".KEY = "Documents"."NewsTopics".TOPIC_ID
```

More than 1 `<tm_return_category>` may be specified so that a document can be categorized with respect to several taxonomies. This is more efficient than using multiple calls because the nearest neighbors of the target document only need to be calculated once.

### Note

**Security Warning:** For this and subsequent text mining functions: The following parameters are SQL expressions. The user application needs to take responsibility for blocking any potentially malicious values from being used.

- `<tm_document> DOCUMENT (<subquery>)`
- `<tm_document> DOCUMENT IN FULLTEXT INDEX WHERE <condition>`
- `<tm_search> ... WHERE <condition>`
- `<tm_search> ... WITH TERM TYPE <string>`, ...

### Description

Text mining supports categorization using the KNN Classifier. No training (up-front preparation) is required because the KNN Classifier uses the reference set documents directly. The KNN Classifier determines the K Nearest Neighbors (most similar documents) from the reference set and then sums and normalizes their similarities per category value to determine the winning category value. If there are multiple categorization columns, then no recomputation is necessary – the same K Nearest Neighbors can determine multiple category columns within the same function call.

The return table contains the suggested categorizations for the target document: The CATEGORY_SCHEMA, CATEGORY_TABLE, CATEGORY_COLUMN columns indicate the given taxonomies, and CATEGORY_LABEL indicates the proposed category. For each categorization proposal, the number of nearest neighbors that opted for the particular category is given, together with a score, which is based on the number and the similarities of the nearest neighbors found.

#### Return table of function TM_CATEGORIZE_KNN

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANK</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CATEGORY_SCHEMA</td>
<td>NVARCHAR(256)</td>
</tr>
<tr>
<td>CATEGORY_TABLE</td>
<td>NVARCHAR(256)</td>
</tr>
<tr>
<td>CATEGORY_COLUMN</td>
<td>NVARCHAR(256)</td>
</tr>
<tr>
<td>CATEGORY_VALUE</td>
<td>NVARCHAR(5000)</td>
</tr>
<tr>
<td>NEIGHBOR_COUNT</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>
### Examples

```sql
SELECT T.CATEGORY_VALUE, T.NEIGHBOR_COUNT, T.SCORE
FROM TM_CATEGORIZE_KNN(
   DOCUMENT ( SELECT filecontent FROM SYSTEM.TM_NEWSAGENCY/Documents WHERE KEY=1 )
   LANGUAGE 'English'
   MIME TYPE 'text/html'
   SEARCH NEAREST NEIGHBORS 22 FILECONTENT
   FROM SYSTEM.TM_NEWSAGENCY/Documents
   WHERE KEY > 100
   WITH TERM TYPE 'Person', 'Organization', 'Location'
   RETURN TOP 3
   REGION
   FROM SYSTEM.TM_NEWSAGENCY/Documents
) AS T
WHERE T.score >= 0.05;
```

```sql
SELECT T.CATEGORY_VALUE, T.NEIGHBOR_COUNT, T.SCORE
FROM TM_CATEGORIZE_KNN(
   DOCUMENT IN FULLTEXT INDEX WHERE KEY=1
   SEARCH NEAREST NEIGHBORS 22 FILECONTENT
   FROM SYSTEM.TM_NEWSAGENCY/Documents
   WHERE KEY > 100
   WITH TERM TYPE 'Person', 'Organization', 'Location'
   RETURN TOP 3
   REGION
   FROM SYSTEM.TM_NEWSAGENCY/Documents
) AS T
WHERE T.score >= 0.05;
```

The following example shows how categorization can be performed if the category information is not stored in the same table as the document text.

```sql
SELECT *
FROM TM_CATEGORIZE_KNN(
   DOCUMENT 'This is the text to be categorized'
   LANGUAGE 'English'
   MIME TYPE 'text/plain'
   SEARCH NEAREST NEIGHBORS 22 FILECONTENT
   FROM SYSTEM.TM_NEWSAGENCY/Documents
   WHERE KEY > 100
   RETURN TOP 3 "topic"
   FROM SYSTEM.TM_NEWSAGENCY/CATEGORIES join SYSTEM.TM_NEWSAGENCY/Documents
) AS T
WHERE T.score >= 0.05;
```
5.4.2 Text Mining: TM_GET_RELATED_DOCUMENTS Function

This text mining function returns the top-ranked related documents for a query document within a search request and stores these documents (including metadata) in the return table.

⚠️ Caution

The information in this topic applies only to the SAP HANA context cited in the title of the topic.

Syntax

```
TM_GET_RELATED_DOCUMENTS ( <tm_document> <tm_search> <tm_return_document> )
```

Specifies the document for which you want to get related documents, the search request and the parameters to be returned with the documents.

Syntax Elements

The function uses the following syntax elements: `<tm_document>`, `<tm_search>`, `<tm_return_document>`, `<tm_document>`::=

```
DOCUMENT { <string> [ LANGUAGE <string> ] [ MIME TYPE <string> ]
    | ( <subquery> ) [ LANGUAGE <string> ] [ MIME TYPE <string> ]
    | IN FULLTEXT INDEX WHERE <condition> }
```

Specifies the document, the languages and the mime types to be used for processing.

There are three ways to specify the query document:

1. Specify the document text as a string. LANGUAGE and MIME TYPE can be specified to indicate how the document is processed. If nothing is specified, the default language (English) and mime type (TEXT/PLAIN) is used.
2. Specify a SELECT query that returns the document text. LANGUAGE and MIME TYPE can be specified as optional parameters.
3. If the query document is part of the full-text index, which was constructed for the set of reference documents, it can be referred to by a WHERE condition. This is much more efficient because the processed document is used immediately.

```
<tm_search> ::= 
    SEARCH <column>
    FROM <table>
    [ WHERE <condition> ]
    [ WITH TERM TYPE <string>, ... ]
```

Specifies the set of reference documents in `<column>` and `<table>`. The specified column must be of type text or must have a full-text index. The set of reference documents can be restricted by WHERE `<condition>`, which means that only those documents are used in the calculations. The set of reference terms can be restricted by WITH TERM TYPE, which means that only those terms are used in the calculations.
For example, WITH TERM TYPE 'proper*', 'noun' only considers terms that are proper names or nouns. The termTypeRestriction property in the text mining configuration determines which term types are available.

```sql
<tm_return_document> ::= RETURN
    [ PRINCIPAL COMPONENTS <pc_int> ] -- output FACTORS, ROTATED_FACTORS
    [ CLUSTERING [ <string> ] ] -- output CLUSTER_LEVEL, CLUSTER_LEFT,
                                  -- CLUSTER_RIGHT
    [ CORRELATION ] -- output CORRELATIONS
    [ HIGHLIGHTED ] -- output HIGHLIGHTED_DOCUMENT,
                       -- HIGHLIGHTED_TERMTYPES
    TOP ( <top_int> | DEFAULT )
    [ <column> [ AS <alias> ], ... ] -- output columns out of <table> in
<tm_search>
```

Specifies the maximum number of related documents to be returned as TOP <top_int>. DEFAULT means to use the value set at text mining initialization. Various options can be specified to request additional information about the found documents and their interrelations.

If PRINCIPAL COMPONENTS <pc_int> is specified, then a principal components analysis (factor analysis) is calculated on the correlation matrix of the found documents. The <top_int> factors are returned as arrays in the FACTORS column of the result table and the rotated factors are returned as ARRAYs in the ROTATED_FACTORS column. If no principal components are requested or if no principal components could be calculated (due to insufficient data or due to linear dependencies in the correlation matrix), then the respective columns of the result table contain NULL values.

If CLUSTERING <string> is specified, then a hierarchical bottom-up cluster analysis is performed on the found related documents. Use <string> to specify an algorithm - 'SINGLE_LINKAGE', 'COMPLETE_LINKAGE', 'AVG_DISTANCE_WITHIN', 'AVG_DISTANCE_BETWEEN', or 'WARD'. The result of the cluster analysis is stored in the CLUSTER_LEVEL, CLUSTER_LEFT, and CLUSTER_RIGHT columns of the result table. The results of these columns can be displayed to the user in the form of a tree diagram. If no cluster analysis was requested or if no cluster analysis could be calculated (due to insufficient data), then these columns contain NULL values.

If CORRELATION is specified, then the correlation matrix between the found documents are returned as arrays in the CORRELATIONS columns of the result table. If this is not specified, then this column contains NULL values.

If HIGHLIGHTED is specified, then the document texts is returned with highlighting information. The HIGHLIGHT_DOCUMENTS column contains the document texts with <span> tags around the most important terms. The HIGHLIGHT_TERMTYPES column contains lists of tag IDs for the different term types. This is useful for highlighting the different term types in different ways.

The HIGHLIGHTED feature is deprecated. It is only available in the legacy text mining implementation, which has been deprecated. For more information, see SAP Note 2435642.

The syntax [ <column> [ AS <alias> ], ... ] specifies a list of columns with possible aliases. The specified columns must be present in the reference table. The values of these columns are returned for the found documents. This avoids the need to have separate calls to retrieve these parameters.

If specified, then the PRINCIPAL COMPONENTS, CLUSTERING, CORRELATION, and HIGHLIGHTED options must be used in this order. TOP must be specified as the last option.
Description

This text mining function returns the top-ranked related documents for a query document within a search request and stores the documents (including metadata) in the return table.

The return table stores the following attributes and additional columns (optional):

Return table of function TM_GETRELATEDDOCUMENTS

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANK</td>
<td>INTEGER</td>
</tr>
<tr>
<td>&lt;column_1&gt;, ..., &lt;column_n&gt;</td>
<td></td>
</tr>
<tr>
<td>TOTAL_TERM_COUNT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>TERM_COUNT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CORRELATIONS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>FACTORS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>ROTATED_FACTORS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>CLUSTER_LEVEL</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>CLUSTER_LEFT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CLUSTER_RIGHT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>HIGHLIGHTEDDOCUMENT</td>
<td>NCLOB</td>
</tr>
<tr>
<td>HIGHLIGHTEDTERMTYPES</td>
<td>NCLOB</td>
</tr>
<tr>
<td>SCORE</td>
<td>DECIMAL</td>
</tr>
</tbody>
</table>

Example

The following example returns the 20 top-ranked related documents for the document John Doe met local president Michael Chang yesterday... since 2014 with the person name Chang in the title from a table that stores news articles.

```
SELECT T.AUTHOR, T.TITLE, T.CREATED_ON, T."content", T.SCORE, T.TOTAL_TERM_COUNT, T.FACTORS[1], T.FACTORS[2], T.CLUSTER_LEVEL
FROM TM_GETRELATEDDOCUMENTS {
  DOCUMENT "John Doe met local president Michael Chang yesterday..."
  SEARCH "content"
  FROM "myschema"."news"
  WHERE CREATED_ON >= '2014-01-01' AND CONTAINS(TITLE, 'Chang')
  WITH TERM TYPE 'PERSON'
  RETURN
  PRINCIPAL COMPONENTS 2
```
5.4.3 Text Mining: TM_GET_RELATED_TERMS Function

This text mining function returns the top-ranked related terms for a query term, based on a set of reference documents.

⚠️ Caution

The information in this topic applies only to the SAP HANA context cited in the title of the topic.

Syntax

```
TM_GET_RELATED_TERMS ( <tm_term> <tm_search> <tm_return_term> )
```

Specifies the term for which you want to find related terms, the search request and the parameters to be returned with the term.

Syntax Elements

The function uses the following syntax elements: `<tm_term>`, `<tm_search>`, `<tm_return_term>`.  

```
<tm_term> ::=  
    TERM <string>  
    [ LANGUAGE <string> ]
```

Specifies the term and the language to be processed. If no language is specified, then the default language (English) is used. The `termColumn` property in the text mining configuration determines which `TA` table column is used for text mining terms. If `termColumn` is set to `TA_NORMALIZED` (the default), then the specified term is normalized to match it. For example, the results for `Mouse` are the same as those for `mouse` in English. "Häuser" and "Haeuser" are also treated as equivalents in German.

The input term string is typically a single term, but in general it can be multiple input terms with optional term types and wildcards. The multiple terms expression evaluates to a simple union of term plus term
type specifications. Internally, text mining combines their term counts as if the terms were conflated in the reference data. The following forms of input term syntax are supported:

The supported forms of input term syntax

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Example</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple term with unspecified (wild-card) term type</td>
<td>sap</td>
<td>&quot;sap*&quot; with any term type</td>
</tr>
<tr>
<td>Term with specified term type</td>
<td>sap:noun</td>
<td>&quot;sap*, but only if used as a noun</td>
</tr>
<tr>
<td>Comma-separated list of terms and optional term types</td>
<td>see, saw:verb, seen</td>
<td>&quot;see&quot; or &quot;seen&quot; (any term type), or &quot;saw&quot; (only as a verb)</td>
</tr>
<tr>
<td>Leading and trailing whitespace (not embedded) is trimmed from each term or term type</td>
<td>... high school ...</td>
<td>&quot;high school&quot; (including embedded space only)</td>
</tr>
<tr>
<td>SQL LIKE wildcards for term or term type</td>
<td>S_P%:ORGANIZATION%</td>
<td>term starting with &quot;S&quot;, followed by any single character, followed by &quot;P&quot;, followed by any additional characters (or none), with term type (entity type) starting with ORGANIZATION (any sub-type); for example, &quot;SAP&quot; or &quot;SAP Inc.&quot;</td>
</tr>
<tr>
<td>Final * wildcard for term or term type (as with termTypeRestriction)</td>
<td>SAP:proper*</td>
<td>&quot;SAP*, term type starting with &quot;proper&quot; (e.g., &quot;proper name&quot;)</td>
</tr>
<tr>
<td>Quote term or term type to allow special characters</td>
<td>'IBM, Inc.':ORGANIZATION*</td>
<td>&quot;IBM, Inc.&quot;, with term type (entity type) starting with ORGANIZATION (any entity sub-type)</td>
</tr>
<tr>
<td>Literal quotes embedded within a quoted string must be doubled</td>
<td>'John O''Brien':PERSON</td>
<td>&quot;John O'Brien&quot; (including space and single-quote, used as the name of a PERSON)</td>
</tr>
<tr>
<td>Term type with unspecified/wildcard term (useful for entities)</td>
<td>*:LOCALITY, :ORGANIZATION%</td>
<td>any entity tagged either as a LOCALITY or as an ORGANIZATION with any sub-type</td>
</tr>
</tbody>
</table>

<tm_search> ::= 
  SEARCH <column> 
  FROM <table> 
  [ WHERE <condition> ] 
  [ WITH TERM TYPE <string>, ... ]

Specifies the set of reference documents in <column> and <table>. The specified column must be of type text or must have a full-text index. The set of reference documents can be restricted by WHERE <condition>, which means that only those documents are used in the calculations. The set of reference terms can be restricted by WITH TERM TYPE, which means that only those terms are used in the calculations.
For example, `WITH TERM TYPE 'proper*','noun'` only considers terms that are proper names or nouns. The `termTypeRestriction` property in the text mining configuration determines which term types are available.

```sql
<tm_return_term> ::=    RETURN       
[ PRINCIPAL COMPONENTS <pc_int> ] -- output FACTORS, ROTATED_FACTORS
[ CLUSTERING [<string>] ] -- output CLUSTER_LEVEL, CLUSTER_LEFT, CLUSTER_RIGHT
[ CORRELATION ] -- output CORRELATIONS
TOP { <top_int> | DEFAULT }
```

Specifies the maximum number of related terms to be returned as `TOP <top_int>`. DEFAULT means to use the value set at text mining initialization. Various options can be specified to request additional information about the found terms and their interrelations.

If `PRINCIPAL COMPONENTS <pc_int>` is specified, then a principal components analysis (factor analysis) is calculated on the correlation matrix of the found terms. The `<top_int>` factors are returned as arrays in the FACTORS column of the result table and the rotated factors will be returned as ARRAYs in the ROTATED_FACTORS column. If no principal components are requested or if no principal components could be calculated (due to insufficient data or due to linear dependencies in the correlation matrix), then the respective columns of the result table contain NULL values.

If `CLUSTERING <string>` is specified, a hierarchical bottom-up cluster analysis will be performed on the found related terms. Use `<string>` to specify algorithm - 'SINGLE_LINKAGE', 'COMPLETE_LINKAGE', 'AVG_DISTANCE_WITHIN', 'AVG_DISTANCE_BETWEEN', or 'WARD'. The result of the cluster analysis is stored in the CLUSTER_LEVEL, CLUSTER_LEFT, and CLUSTER_RIGHT columns of the result table. The results of these columns can be displayed to the user in the form of a tree diagram. If no cluster analysis was requested or if no cluster analysis could be calculated (due to insufficient data), then these columns contain NULL values.

If `CORRELATION` is specified, then the correlation matrix between the found terms will be returned as arrays in the CORRELATIONS column of the result table. If this is not specified, then this column contains NULL values.

If specified, then the options `PRINCIPAL COMPONENTS`, `CLUSTERING`, and `CORRELATION` must be used in this order. TOP must always be specified as the last option.

**Description**

This text mining function returns the top-ranked related terms for a query term, based on a set of reference documents. The return table contains the terms found together with additional information about the terms and their interrelation. The calculation of related terms is based on co-occurrence.

The return table stores the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANK</td>
<td>INTEGER</td>
</tr>
<tr>
<td>TERM</td>
<td>NVARCHAR(2000)</td>
</tr>
</tbody>
</table>
### Attribute and Value Data Type

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMALIZED_TERM</td>
<td>NVARCHAR(2000)</td>
</tr>
<tr>
<td>TERM_TYPE</td>
<td>NVARCHAR(256)</td>
</tr>
<tr>
<td>TERM_FREQUENCY</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DOCUMENT_FREQUENCY</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CORRELATIONS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>FACTORS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>ROTATED_FACTORS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>CLUSTER_LEVEL</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>CLUSTER_LEFT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CLUSTER_RIGHT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>SCORE</td>
<td>DECIMAL</td>
</tr>
</tbody>
</table>

**This column contains the total number of times a term occurred in the specified reference documents.**

**This column contains the number of reference documents in which a term occurred.**

### Example

The following example returns the 12 top-ranked related terms for `baseball` of type `person` and `organization` from a table that stores news articles.

```sql
SELECT T.RANK, T.TERM, T.NORMALIZED_TERM, T.TERM_TYPE, T.TERM_FREQUENCY, T.DOCUMENT_FREQUENCY, T.SCORE
FROM TM_GET_RELATED_TERMS (TERM 'baseball'
    LANGUAGE 'english'
    SEARCH "content"
    FROM "myschema"."news"
    WHERE CREATED_ON >='2014-01-01'
    WITH TERM TYPE 'PERSON', 'ORGANIZATION'
RETURN TOP 12
) AS T
WHERE T.SCORE > 0.1;
```
### 5.4.4 Text Mining: TM_GET_RELEVANT_DOCUMENTS Function

This text mining function returns the top-ranked documents that are relevant to a term.

⚠️ Caution

The information in this topic applies only to the SAP HANA context cited in the title of the topic.

#### Syntax

```
TM_GET_RELEVANT_DOCUMENTS ( <tm_term> <tm_search> <tm_return_document> )
```

Specifies the term for which you want to find relevant documents, the search request and the parameters to be returned with the documents.

#### Syntax Elements

The function uses the following syntax elements: `<tm_term>`, `<tm_search>`, `<tm_return_document>`.  

```
<tm_term> ::=      TERM <string>     
              [ LANGUAGE <string> ]
```

Specifies the term and the language to be processed. If no language is specified, then the default language (English) is used. The `termColumn` property in the text mining configuration determines which $TA table column is used for text mining terms. If `termColumn` is set to `TA_NORMALIZED` (the default), then the specified term is normalized to match it. For example, the results for `Mouse` are the same as those for `mouse` in English. **Häuser** and **Haeuser** are also treated as equivalents in German.
The input term string is typically a single term, but in general it can be multiple input terms with optional term types and wildcarding. See the description of `TM_GET_RELEVANT_TERMS` for details.

```sql
<tm_search> ::= 
    SEARCH <column> 
    FROM <table> 
    [ WHERE <condition> ] 
    [ WITH TERM TYPE <string>, ... ]
```

Specifies the set of reference documents in `<column>` and `<table>`. The specified column must be of type text or must have a full-text index. The set of reference documents can be restricted by `WHERE <condition>`, which means that only those documents are used in the calculations. The set of reference terms can be restricted by `WITH TERM TYPE`, which means that only those terms are used in the calculations. For example `WITH TERM TYPE 'proper*','noun'` only considers terms that are proper names or nouns. The termTypeRestriction property in the text mining configuration determines which term types are available.

```sql
<tm_return_document> ::= 
    RETURN 
    [ PRINCIPAL COMPONENTS <pc_int> ] -- output FACTORS, ROTATED_FACTORS 
    [ CLUSTERING ['string']]        -- output CLUSTER_LEVEL, CLUSTER_LEFT, 
                      -- CLUSTER_RIGHT 
    [ CORRELATION ]                -- output CORRELATIONS 
    [ HIGHLIGHTED ]                -- output HIGHLIGHTED_DOCUMENT, 
                      -- HIGHLIGHTED_TERMTYPES 
    TOP ( <top_int> | DEFAULT )    -- output columns out of <table> in 
    [ <column> [ AS <alias> ], ... ]
```

Specifies the maximum number of relevant documents to be returned as `TOP <top_int>`. DEFAULT means to use the value set at text mining initialization. Various options can be specified to request additional information about the found documents and their interrelations.

If `PRINCIPAL COMPONENTS <pc_int>` is specified, then a principal components analysis (factor analysis) is calculated on the correlation matrix of the found documents. The `<top_int>` factors are returned as arrays in the column FACTORS of the result table and the rotated factors are returned as ARRAYS in the ROTATED_FACTORS column. If no principal components are requested or if no principal components could be calculated (due to insufficient data or due to linear dependencies in the correlation matrix), then the respective columns of the result table contain NULL values.

If `CLUSTERING ['string]` is specified, then a hierarchical bottom-up cluster analysis is performed on the found documents. Use `['string]` to specify algorithm - ‘SINGLE_LINKAGE’, ‘COMPLETE_LINKAGE’, ‘AVG_DISTANCE_WITHIN’ , ‘AVG_DISTANCE_BETWEEN’, or ‘WARD’. The result of the cluster analysis is stored in the CLUSTER_LEVEL, LUSTER_LEFT, and CLUSTER_RIGHT columns of the result table. The results of these columns can be displayed to the user in the form of a tree diagram. If no cluster analysis was requested or if no cluster analysis could be calculated (due to insufficient data), then these columns contain NULL values.

If `CORRELATION` is specified, then the correlation matrix between the found terms is returned as arrays in the CORRELATIONS column of the result table. If this is not specified, then this column contains NULL values.

If `HIGHLIGHTED` is specified, then the document texts together with highlighting information is returned. The HIGHLIGHT/Documents column contains the document texts with `span` tags around the most important terms. The HIGHLIGHT_TERMTYPES column contains lists of tag IDs for the for the different term types. This is useful for highlighting the different term types in different ways.

The HIGHLIGHTED feature is deprecated. It is only available in the legacy text mining implementation, which has been deprecated. For more information, see SAP Note 2435642.
The syntax `[ <column> [ AS <alias> ], ... ]` specifies a list of columns with possible aliases. The specified columns must be present in the reference table. The values of these columns are returned for the found documents. This avoids the need to have separate calls to retrieve these parameters.

If specified, then the PRINCIPAL COMPONENTS, CLUSTERING, CORRELATION, and HIGHLIGHTED options must be used in this order. TOP must always be specified as the last option.

**Description**

This text mining function returns the top-ranked documents that are relevant to a query term within a search request, and stores the documents (including metadata) in the return table.

The return table stores the following attributes and additional columns (optional):

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANK</td>
<td>INTEGER</td>
</tr>
<tr>
<td><code>&lt;column_1&gt;, ..., &lt;column_n&gt;</code></td>
<td></td>
</tr>
<tr>
<td>TOTAL_TERM_COUNT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>TERM_COUNT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CORRELATIONS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>FACTORS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>ROTATED_FACTORS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>CLUSTER_LEVEL</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>CLUSTER_LEFT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CLUSTER_RIGHT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>HIGHLIGHTED_DOCUMENT</td>
<td>NCLOB</td>
</tr>
<tr>
<td>HIGHLIGHTED_TERMTYPES</td>
<td>NCLOB</td>
</tr>
<tr>
<td>SCORE</td>
<td>DECIMAL</td>
</tr>
</tbody>
</table>

**Example**

The following example returns the 20 top-ranked related documents, which include the term/person John Doe from a table that stores news articles.

```sql
SELECT T.AUTHOR, T.TITLE, T.CREATED_ON, T.SCORE
```
FROM TM_GET_RELEVANT_DOCUMENTS (  
  TERM 'John Doe'  
  LANGUAGE 'english'  
  SEARCH "content"  
  FROM "myschema"."news"  
  WHERE CREATED_ON >= '2014-01-01'  
  RETURN  
  TOP 20  
  AUTHOR, TITLE, CREATED_ON  
) AS T  
WHERE T.SCORE > 0.25 AND T.TOTAL_TERM_COUNT > 3;

5.4.5 Text Mining: TM_GET_RELEVANT_TERMS Function

This text mining function returns the top-ranked relevant terms that describe a document.

⚠️ Caution

The information in this topic applies only to the SAP HANA context cited in the title of the topic.

Syntax

```
TM_GET_RELEVANT_TERMS ( <tm_document> <tm_search> <tm_return_term> )
```

Specifies the document for which you want to find relevant terms, the search request and the parameters to be returned with the term.

Syntax Elements

The function uses the following syntax elements: `<tm_document>`, `<tm_search>`, `<tm_return_term>`.

```
<tm_document> ::=  
  DOCUMENT { <string> [ LANGUAGE <string> ] [ MIME TYPE <string> ]  
  | ( <subquery> ) [ LANGUAGE <string> ] [ MIME TYPE <string> ]  
  | IN FULLTEXT INDEX WHERE <condition> }
```

Specifies the document for which you want to find the most relevant terms.

There are three ways to specify the query document:

1. Specify the document text as a string. LANGUAGE and MIME TYPE can be specified to indicate how the document is processed. If nothing is specified, then the default language (English) and mime type (TEXT/PLAIN) are used.
2. Specify a SELECT query that returns the document text. LANGUAGE and MIME TYPE can be specified as optional parameters.
3. If the query document is part of the full-text index that was constructed for the set of reference
documents, then it can be referred to by a WHERE condition. This is much more efficient since the
processed document is used immediately.

```sql
<tm_search> ::= 
SEARCH <column> 
FROM <table> 
[ WHERE <condition> ] 
[ WITH TERM TYPE <string>, ... ]
```

Specifies the set of reference documents in `<column>` and `<table>`. The specified column must be of type
text or must have a full-text index. The set of reference documents can be restricted by WHERE `<condition>`.
which means that only those documents are used in the calculations. The set of reference terms can
be restricted by WITH TERM TYPE, which means that only those terms are used in the calculations. For
example, WITH TERM TYPE 'proper*', 'noun' only considers terms that are proper names or nouns. The
termTypeRestriction property in the text mining configuration determines which term types are available.

```sql
<tm_return_term> ::= 
RETURN 
[ PRINCIPAL COMPONENTS <pc_int> ] -- output FACTORS, ROTATED_FACTORS 
[ CLUSTERING [ <string> ] ] -- output CLUSTER_LEVEL, CLUSTER_LEFT, CLUSTER_RIGHT 
[ CORRELATION ] -- output CORRELATIONS 
TOP { <top_int> | DEFAULT }
```

Specifies the maximum number of relevant terms to be returned as TOP `<top_int>`. DEFAULT means to use
the value set at text mining initialization. Various options can be specified to request additional information
about the found terms and their interrelations.

If PRINCIPAL COMPONENTS `<pc_int>` is specified, then a principal components analysis (factor analysis) is
calculated on the correlation matrix of the found terms. The `<pc_int>` factors are returned as arrays in the
column FACTORS of the result table and the rotated factors are returned as ARRAYS in the ROTATED_FACTORS
column. If no principal components are requested or if no principal components could be calculated (due to
insufficient data or due to linear dependencies in the correlation matrix), then the respective columns of the
result table contain NULL values.

If CLUSTERING `<string>` is specified, a hierarchical bottom-up cluster analysis is performed on the
found relevant terms. Use `<string>` to specify algorithm - 'SINGLE_LINKAGE', 'COMPLETE_LINKAGE',
'AVG_DISTANCE_WITHIN', 'AVG_DISTANCE_BETWEEN', or 'WARD'. The result of the cluster analysis is stored
in the CLUSTER_LEVEL, CLUSTER_LEFT, and CLUSTER_RIGHT columns of the result table. The results of
these columns can be displayed to the user in the form of a tree diagram. If no cluster analysis was requested
or if no cluster analysis could be calculated (due to insufficient data), then these columns contain NULL values.

If CORRELATION is specified, then the correlation matrix between the found terms are returned as arrays in
the CORRELATIONS column of the result table. If this is not specified, then this column contains NULL values.

If specified, then the PRINCIPAL COMPONENTS, CLUSTERING, and CORRELATION options must be used in
this order. TOP must always be specified as the last option.

Description

This text mining function returns the top-ranked relevant terms that describe a document.
The return table stores the following attributes:

Return table of function TM_GET_RELEVANT_TERMS

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANK</td>
<td>INTEGER</td>
</tr>
<tr>
<td>TERM</td>
<td>NVARCHAR(2000)</td>
</tr>
<tr>
<td>NORMALIZED_TERM</td>
<td>NVARCHAR(2000)</td>
</tr>
<tr>
<td>TERM_TYPE</td>
<td>NVARCHAR(256)</td>
</tr>
<tr>
<td>TERM_FREQUENCY</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DOCUMENT_FREQUENCY</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CORRELATIONS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>FACTORS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>ROTATED_FACTORS</td>
<td>DECIMAL ARRAY</td>
</tr>
<tr>
<td>CLUSTER_LEVEL</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>CLUSTER_LEFT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>CLUSTER_RIGHT</td>
<td>INTEGER</td>
</tr>
<tr>
<td>SCORE</td>
<td>DECIMAL</td>
</tr>
</tbody>
</table>

**Example**

The following example returns the 20 top-ranked relevant terms for the document with the DOC_ID 22 from a table that stores news articles.

```sql
SELECT *
FROM TM_GET_RELEVANT_TERMS (    DOCUMENT (SELECT content FROM "myschema"."news" WHERE DOC_ID = 22)
    SEARCH "content"
    FROM "myschema"."news"
    WHERE CREATED_ON >= '2014-01-01'
RETURN
    PRINCIPAL COMPONENTS 2
    CLUSTERING
    TOP 20
) AS T
WHERE T.SCORE > 0.1;
```
5.4.6 Text Mining: TM_GET_SUGGESTED_TERMS Function

This text mining function returns the top-ranked terms that match an initial substring. This function can be used for type-ahead or auto-completion functions.

⚠️ Caution
The information in this topic applies only to the SAP HANA context cited in the title of the topic.

Syntax

```sql
TM_GET_SUGGESTED_TERMS ( <tm_term> <tm_search> <tm_return_top> )
```

Specifies the initial substring for which you want to find suggested terms, the search request and the number of returned terms.

Syntax Elements

The function uses the following syntax elements: `<tm_term>`, `<tm_search>`, `<tm_return_top>`.

- `<tm_term>` ::= TERM <string> [ LANGUAGE <string> ]
  Specifies the term and the language to be processed. If no language is specified, then the default language (English) is used. The `termColumn` property in the text mining configuration determines which $TA table column is used for text mining terms. If `termColumn` is set to `TA_NORMALIZED` (the default), then the specified term is normalized to match it. For example, the results for `Mouse` are the same as those for `mouse` in English. `Häuser` and `Haeuser` are also treated as equivalents in German.

- `<tm_search>` ::= SEARCH <column> FROM <table> [ WHERE <condition> ] [ WITH TERM TYPE <string>, ... ]
  Specifies the set of reference documents in `<column>` and `<table>`. The specified column must be of type text or must have a full-text index. The set of reference documents can be restricted by WHERE `<condition>`. The set of reference terms can be restricted by WITH TERM TYPE, which means that only those documents are used in the calculations. The set of reference terms can be restricted by WITH TERM TYPE, which means that only those terms are used in the calculations. For example, `WITH TERM TYPE 'proper','noun'` only considers terms that are proper names or nouns. The `termTypeRestriction` property in the text mining configuration determines which term types are available.

- `<tm_return_top>` ::= RETURN TOP { <top_int> | DEFAULT }
  Specifies the number of top returned terms. DEFAULT means to use the value set at text mining initialization.
Description

This text mining function returns the top-ranked terms that match an initial substring. You might know this behavior from the type-ahead function or auto-completion within (search) input fields.

Return table of function TM_GET_SUGGESTED_TERMS

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANK</td>
<td>INTEGER</td>
</tr>
<tr>
<td>TERM</td>
<td>NVARCHAR(2000)</td>
</tr>
<tr>
<td>NORMALIZED_TERM</td>
<td>NVARCHAR(2000)</td>
</tr>
<tr>
<td>TERM_TYPE</td>
<td>NVARCHAR(256)</td>
</tr>
<tr>
<td>TERM_FREQUENCY</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DOCUMENT_FREQUENCY</td>
<td>INTEGER</td>
</tr>
<tr>
<td>SCORE</td>
<td>DECIMAL</td>
</tr>
</tbody>
</table>

Example

The following example returns terms starting with *john* from a news table. Single-word terms and multi-word terms are returned.

```sql
SELECT T.RANK, T.TERM, T.NORMALIZED_TERM, T.TERM_TYPE, T.TERM_FREQUENCY,
       T.DOCUMENT_FREQUENCY, T.SCORE
FROM TM_GET_SUGGESTED_TERMS (TERM 'john'
                                SEARCH "content"
                                FROM "myschema"."news"
                                RETURN
                                TOP 5)
       AS T
WHERE T.SCORE > 0.1 AND T.TERM_FREQUENCY > 5;
```

Example Return Table

<table>
<thead>
<tr>
<th>RANK</th>
<th>TERM</th>
<th>NORMALIZED_TERM</th>
<th>TERM_TYPE</th>
<th>TERM_FREQUENCY</th>
<th>DOCUMENT_FREQUENCY</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>john</td>
<td>Proper</td>
<td>25</td>
<td>11</td>
<td>0.86</td>
</tr>
<tr>
<td>2</td>
<td>John Doe</td>
<td>john doe</td>
<td>Proper</td>
<td>12</td>
<td>8</td>
<td>0.24</td>
</tr>
<tr>
<td>3</td>
<td>John Miller</td>
<td>john miller</td>
<td>Proper</td>
<td>5</td>
<td>3</td>
<td>0.21</td>
</tr>
<tr>
<td>RANK</td>
<td>TERM</td>
<td>NORMALIZED_TERM</td>
<td>TERM_TYPE</td>
<td>TERM_FREQUENCY</td>
<td>DOCUMENT_FREQUENCY</td>
<td>SCORE</td>
</tr>
<tr>
<td>------</td>
<td>---------------</td>
<td>-----------------</td>
<td>------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>-------</td>
</tr>
<tr>
<td>4</td>
<td>Johny</td>
<td>johnny</td>
<td>Proper</td>
<td>7</td>
<td>7</td>
<td>0.15</td>
</tr>
<tr>
<td>5</td>
<td>Johnson and Sons</td>
<td>johnson and son</td>
<td>Organization</td>
<td>5</td>
<td>2</td>
<td>0.09</td>
</tr>
</tbody>
</table>
6 System Views Reference

SAP HANA includes system views, monitoring views, and embedded statistics service views.

**System Views [page 1476]**
System views allow you to query for various information about the system state using SQL commands. The results appear as tables.

**Monitoring Views [page 1729]**
SAP HANA includes a set of runtime views called monitoring views that provide actual SAP HANA runtime data, including statistics and status information related to the execution of DML statements. These views are useful for monitoring and troubleshooting performance. The data in monitoring views is not stored on disk; it is calculated when you execute a query on one of the views.

Related Information

System Views [page 1476]
Monitoring Views [page 1729]
Embedded Statistics Service Views (_SYS_STATISTICS schema)

6.1 System Views

System views allow you to query for various information about the system state using SQL commands. The results appear as tables.

System views are located in the SYS schema. In a system with tenant databases, every database has a SYS schema with system views that contain information about that database only.

In addition, the system database has a SYS_DATABASES schema that has corresponding proxy schemas (SYS_DATABASES_SR_SITE_secondary_site_name) that collectively provide aggregated system information for all tenant databases in the system. These views all have the column, DATABASE_NAME, which allows you to identify which database the information refers to. To view information in these views, you need the CATALOG READ or DATABASE ADMIN system privilege.

**Time-related columns in views**: Time-related columns in system views contain data in the local time (not UTC) unless specifically noted in the description for the column.

**Metadata views vs runtime views**: SAP HANA system views are separated into two categories: metadata views and runtime views. Metadata views provide metadata about objects in the database, including options or settings that were set using a DDL statement. Runtime views provide actual HANA runtime data, including statistics and status information related to the execution of DML statements. Runtime views start with M_ for monitoring.
Normally, the names are symmetric - for example, TABLES and M_TABLES, and they may also include some other defining abbreviation, such as INDEXES versus M_CS_INDEXES, and so on.

**Catalog information for Statistics Services:** For information on views that support the statistics services feature starting with HANA 1.0 SPS 07, see the topic on Embedded Statistics Service Views (_SYS_STATISTICS schema).

### 6.1.1 ABSTRACT_SQL_PLANS System View

Lists information about abstract SQL plans.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUERY_ID</td>
<td>BIGINT</td>
<td>Displays a number that uniquely identifies the abstract SQL query.</td>
</tr>
<tr>
<td>ABSTRACT_SQL_PLAN_ID</td>
<td>BIGINT</td>
<td>Displays a number that uniquely identifies each abstract SQL plan entry.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name of the location where the abstract SQL plan was captured.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port number of the location where the abstract SQL plan was captured.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the volume where the abstract SQL plan was captured.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the target statement string.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the MD5 hash value for the STATEMENT_STRING column.</td>
</tr>
<tr>
<td>PLAN_KEY</td>
<td>NVARCHAR(5000)</td>
<td>Displays the key for the abstract SQL plan in JSON format. These values can affect the generated query execution plans.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ABSTRACT_SQL_PLAN</td>
<td>NCLOB</td>
<td>Displays the abstract SQL plan in JSON format. For example:</td>
</tr>
</tbody>
</table>
|                     |           | `{"alp_rels":[{"alp_Rel": {"alias":"DUMMY","alp_id":1,"enumerated_by":139,"field_names":[]}, "hash_partition_column":-2147483648,"input_size":1,"intermediate_alternative":false,"is_applied":false,"is_esx_node":false,"is_view_cache":false,"join_table_name":"NULL","locationName":"selibm57:36903","locked":false,"logicallyEnumerated":false,"multi_container_id":-1,"output_column_size":1,"partition_search_ids":[]},"physical_operator_type":22,"referenced_cols": [{"col":0}],"rel_id":1000000,"rel_type":6,"schema_name":"SYS","table_name":"DUMMY","table_type":0,"trex_externalkey_pos":-2147483648,"trex_rel_id":-1,"trex_rowid_needed":false,"trex_rowid_pos":-2147483648,"used_cols":{"col":0}},{"alp_Rel": {"alp_id":0,"child_rel_id":1,"enumerated_by":139,"intermediate_alternative":false,"is_esx_node":false,"locationName":"selibm57:36903","locked":false,"logicallyEnumerated":true,"output_column_size":0,"physical_operator_type":0,"project_col_aliases": [{"alias":"NULL"}],"project_col_labels": [{"label":"DUMMY"}],"project_cols": [{"alp_Exp": {"col_id":0,"exp_type":0,"field_name":"DUMMY","is_grouping_id":false,"is_table_key":false,"org_table_name":"DUMMY","position":-1,"prefetch":false,"promoted_type": {"ftc":0,"length":-1,"scale":0},"real_result_type": {"ftc":8,"length":1,"scale":0},"rel_id":1000000,"result_type": {"ftc":29,"length":1,"scale":0},"schema_name":"SYS","table_name":"DUMMY"}],"rel_type":3,"trex_rel_id":-1}]}],"applied_migrator_set": ["3.0","4.0","5.0"],"related_objects": [{"object_id":132355,"object_name":"SYS.DUMMY","object_type":1,"object_version":1}],"root_rel_id":0}`
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATED_OBJECTS</td>
<td>NCLOB</td>
<td>Displays related object information for the captured abstract SQL plan.</td>
</tr>
<tr>
<td>ABSTRACT_SQL_PLAN_VERSION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the abstract SQL plan version with applied migration information.</td>
</tr>
<tr>
<td>IS_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the abstract SQL plan will be used (TRUE) or not (FALSE).</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the abstract SQL plan is valid (TRUE) or invalid (FALSE).</td>
</tr>
<tr>
<td>DETAILS</td>
<td>NCLOB</td>
<td>Displays additional information on the abstract SQL plan, including invalidation reasons.</td>
</tr>
<tr>
<td>LAST_ENABLE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the abstract SQL plan was added or last enabled.</td>
</tr>
<tr>
<td>LAST_ENABLE_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the ID of the user who added or last enabled the abstract SQL plan.</td>
</tr>
<tr>
<td>LAST_DISABLE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the abstract SQL plan was last disabled.</td>
</tr>
<tr>
<td>LAST_DISABLE_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the ID of the user who last disabled the abstract SQL plan.</td>
</tr>
<tr>
<td>CAPTURE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the abstract SQL plan was captured.</td>
</tr>
<tr>
<td>CAPTURE_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the ID of the user who captured the abstract SQL plan.</td>
</tr>
</tbody>
</table>

**Related Information**

- ALTER SYSTEM {ENABLE | DISABLE | REMOVE} ABSTRACT SQL PLAN (System Management) [page 532]
- ALTER SYSTEM MIGRATE ABSTRACT SQL PLAN (System Management) [page 544]
- ALTER SYSTEM [START | STOP] CAPTURE ABSTRACT SQL PLAN (System Management) [page 575]
- ALTER SYSTEM [START | STOP] APPLY ABSTRACT SQL PLAN (System Management) [page 570]
- ALTER SYSTEM UPDATE ABSTRACT SQL PLAN Statement (System Management) [page 586]
- M_ABSTRACT_SQL_PLAN_OVERVIEW System View [page 1730]
- M_ABSTRACT_SQL_PLAN_STATISTICS System View [page 1731]
- SQL Plan Stability
6.1.2 ACCESSIBLE_VIEWS System View

Provides information about accessible views for a given user.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user who is authorized to access the view.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the view.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the view.</td>
</tr>
<tr>
<td>ANALYTICAL_PRIVILEGE_NEEDED</td>
<td>VARCHAR(5)</td>
<td>Displays whether this view is secured by means of analytical privileges.</td>
</tr>
</tbody>
</table>

Additional Information

This view requires an equal predicate on USER_NAME.

Related Information

System Views for Verifying Users' Authorization

6.1.3 ADAPTERS System View

Displays adapters available in the SAP HANA system.

Structure

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(64)</td>
<td>Displays the adapter name.</td>
</tr>
</tbody>
</table>
### Related Information

ADAPTER_CAPABILITIES System View [page 1481]
ADAPTER_LOCATIONS System View [page 1482]
Create an SAP HANA Remote Source

# 6.1.4 ADAPTER_CAPABILITIES System View

Displays supported capabilities for each adapter.

## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(64)</td>
<td>Displays the adapter name.</td>
</tr>
<tr>
<td>SOURCE_VERSION</td>
<td>NVARCHAR(64)</td>
<td>Displays the source versions that are supported by the adapter.</td>
</tr>
<tr>
<td>CAPABILITY_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the capability name.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>NVARCHAR(512)</td>
<td>Displays the description of the capability.</td>
</tr>
<tr>
<td>SCOPE</td>
<td>VARCHAR(10)</td>
<td>Displays the capability scope.</td>
</tr>
<tr>
<td>IS_SDA_SUPPORTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the SDA capability is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>IS_CDC_SUPPORTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the CDC capability is supported: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

ADAPTERS System View [page 1480]
ADAPTER_LOCATIONS System View [page 1482]

### 6.1.5 ADAPTER_LOCATIONS System View

Displays the location of adapters.

**Structure**

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(64)</td>
<td>Displays the adapter name.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>VARCHAR(11)</td>
<td>Displays the location of the adapter: 'indexserver', 'dpserver', 'agent'.</td>
</tr>
<tr>
<td>AGENT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the agent name.</td>
</tr>
<tr>
<td>CONFIGURATION</td>
<td>NCLOB</td>
<td>Displays the configuration of the adapter.</td>
</tr>
<tr>
<td>PROPERTIES</td>
<td>NCLOB</td>
<td>Displays the properties of the adapter.</td>
</tr>
</tbody>
</table>

Related Information

ADAPTERS System View [page 1480]
ADAPTER_CAPABILITIES System View [page 1481]
6.1.6 AFL_AREAS System View

Provides information about available AFL areas.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREA_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the AFL area.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the AFL area.</td>
</tr>
<tr>
<td>AREA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the AFL area.</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the AFL area was created.</td>
</tr>
</tbody>
</table>

Related Information

AFL_FUNCTIONS System View [page 1484]
AFL_FUNCTION_PARAMETERS System View [page 1485]
AFL_FUNCTION_PROPERTIES System View [page 1486]
AFL_PACKAGES System View [page 1487]
AFL_TEXTS System View [page 1488]
M_AFL_STATES System View [page 1743]
M_PLUGIN_STATUS System View [page 2054]
CREATE WORKLOAD MAPPING Statement (Workload Management) [page 934]
ALTER WORKLOAD MAPPING Statement (Workload Management) [page 681]
6.1.7 AFL_FUNCTIONS System View

Provides information about available AFL functions.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTION_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the AFL function.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the AFL function.</td>
</tr>
<tr>
<td>AREA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the AFL area that the AFL function belongs to.</td>
</tr>
<tr>
<td>PACKAGE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the AFL package that the AFL function belongs to.</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the AFL function.</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the AFL function was created.</td>
</tr>
<tr>
<td>INPUT_PARAMETER_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of the input parameters of the AFL function.</td>
</tr>
<tr>
<td>RETURN_VALUE_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of the output parameters of the AFL function.</td>
</tr>
<tr>
<td>FUNCTION_TYPE</td>
<td>VARCHAR(7)</td>
<td>Displays the type of the AFL function.</td>
</tr>
<tr>
<td>TECHNICAL_CATEGORY</td>
<td>VARCHAR(11)</td>
<td>Displays the technical category of the AFL function.</td>
</tr>
</tbody>
</table>

Related Information

- AFL_AREAS System View [page 1483]
- AFL_FUNCTION_PARAMETERS System View [page 1485]
- AFL_FUNCTION_PROPERTIES System View [page 1486]
- AFL_PACKAGES System View [page 1487]
- AFL_TEXTS System View [page 1488]
- M_AFL_STATES System View [page 1743]
6.1.8 AFL_FUNCTION_PARAMETERS System View

Provides information about parameters of AFL functions.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the AFL function.</td>
</tr>
<tr>
<td>AREA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the area name of the AFL function.</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the function name of the AFL function.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the parameter name.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the parameter.</td>
</tr>
<tr>
<td>IO_TYPE</td>
<td>NVARCHAR(8)</td>
<td>Displays the direction of the parameter.</td>
</tr>
<tr>
<td>DATA_TYPE</td>
<td>VARCHAR(22)</td>
<td>Displays the data type.</td>
</tr>
<tr>
<td>CS_DATA_TYPE</td>
<td>VARCHAR(13)</td>
<td>Displays the column store data type.</td>
</tr>
<tr>
<td>CS_DATA_TYPE_EXPRESSION</td>
<td>NVARCHAR(256)</td>
<td>Displays the regular expression of column store data type.</td>
</tr>
<tr>
<td>SQL_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the SQL name.</td>
</tr>
<tr>
<td>SQL_DATA_TYPE</td>
<td>VARCHAR(14)</td>
<td>Displays the SQL data type.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the parameter length in bytes.</td>
</tr>
<tr>
<td>RAW_SIZE</td>
<td>INTEGER</td>
<td>Displays the parameter raw size in bytes.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Displays the scale of the parameter.</td>
</tr>
<tr>
<td>USAGE</td>
<td>NVARCHAR(32)</td>
<td>Displays the usage.</td>
</tr>
<tr>
<td>BUSINESS_TEXT_ID</td>
<td>INTEGER</td>
<td>Displays the business text ID of the parameter.</td>
</tr>
</tbody>
</table>
### Related Information

- AFL_FUNCTIONS System View [page 1484]
- AFL_PACKAGES System View [page 1487]
- AFL_FUNCTION_PROPERTIES System View [page 1486]
- AFL_TEXTS System View [page 1488]
- M_AFL_STATES System View [page 1743]

### 6.1.9 AFL_FUNCTION_PROPERTIES System View

Provides information about available AFL function properties.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema of the AFL function properties.</td>
</tr>
<tr>
<td>AREA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the area name of the AFL function properties.</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the function name of the AFL function properties.</td>
</tr>
<tr>
<td>KEY</td>
<td>NVARCHAR(256)</td>
<td>Displays the key of the AFL function properties.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(1024)</td>
<td>Displays the value of the AFL function properties.</td>
</tr>
</tbody>
</table>

### Related Information

- AFL_AREAS System View [page 1483]
- AFL_FUNCTIONS System View [page 1484]
- AFL_FUNCTION_PARAMETERS System View [page 1485]
- AFL_PACKAGES System View [page 1487]
- AFL_TEXTS System View [page 1488]
6.1.10 AFL_PACKAGES System View

Provides information about available AFL packages.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACKAGE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the AFL package.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the AFL package.</td>
</tr>
<tr>
<td>AREA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the AFL area that the AFL package belongs to.</td>
</tr>
<tr>
<td>PACKAGE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the AFL package.</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the AFL package was created.</td>
</tr>
</tbody>
</table>

Related Information

- AFL_AREAS System View [page 1483]
- AFL_FUNCTIONS System View [page 1484]
- AFL_FUNCTION_PARAMETERS System View [page 1485]
- AFL_FUNCTION_PROPERTIES System View [page 1486]
- AFL_TEXTS System View [page 1488]
6.1.11 AFL_TEXTS System View

Provides information about available AFL texts.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the AFL text.</td>
</tr>
<tr>
<td>AREA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the area name of the AFL text.</td>
</tr>
<tr>
<td>TEXT_ID</td>
<td>INTEGER</td>
<td>Displays the text ID.</td>
</tr>
<tr>
<td>LANGUAGE_CODE</td>
<td>NVARCHAR(5)</td>
<td>Displays the language code of the AFL text.</td>
</tr>
<tr>
<td>TEXT</td>
<td>NVARCHAR(512)</td>
<td>Displays the text.</td>
</tr>
</tbody>
</table>

**Related Information**

- AFL_AREAS System View [page 1483]
- AFL_FUNCTIONS System View [page 1484]
- AFL_FUNCTION_PARAMETERS System View [page 1485]
- AFL_FUNCTION_PROPERTIES System View [page 1486]
- AFL_PACKAGES System View [page 1487]

6.1.12 AGENTS System View

Lists active data provisioning agents in the system.

**Structure**

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGENT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the agent name.</td>
</tr>
</tbody>
</table>
### Column Data type Description

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTOCOL</td>
<td>VARCHAR(4)</td>
<td>Displays the protocol for communication with SAP HANA database: TCP/HTTP.</td>
</tr>
<tr>
<td>AGENT_HOST</td>
<td>NVARCHAR(64)</td>
<td>Displays the agent host specified when using TCP.</td>
</tr>
<tr>
<td>AGENT_PORT</td>
<td>INTEGER</td>
<td>Displays the agent port number specified when using TCP.</td>
</tr>
<tr>
<td>IS_SSL_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the agent listening on the TCP port uses SSL: TRUE/FALSE.</td>
</tr>
<tr>
<td>AGENT_GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the agent clustering group to which the agent belongs.</td>
</tr>
</tbody>
</table>

### Related Information

- AGENT_CONFIGURATION System View [page 1489]
- AGENT_GROUPS System View [page 1490]
- M_AGENTS System View [page 1744]

### 6.1.13 AGENT_CONFIGURATION System View

Provides agent configuration information.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGENT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the agent name.</td>
</tr>
<tr>
<td>KEY</td>
<td>VARCHAR(128)</td>
<td>Displays the agent property key.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NCLOB</td>
<td>Displays the agent property value.</td>
</tr>
</tbody>
</table>

### Related Information

- AGENTS System View [page 1488]
- AGENT_GROUPS System View [page 1490]
6.1.14 AGENT_GROUPS System View

Lists active data provisioning agent groups in the system.

**Structure**

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGENT_GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the agent group.</td>
</tr>
</tbody>
</table>

**Related Information**

AGENTS System View [page 1488]
AGENT_CONFIGURATION System View [page 1489]
M_AGENTS System View [page 1744]

6.1.15 ALL_AUDIT_LOG System View

Provides information about audit records, including those for XSA events. You must have AUDIT ADMIN, AUDIT OPERATOR, or AUDIT READ system privileges to access this view.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the time that the event occurred.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the name of the host where the event occurred.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port number.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the name of the service.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>CLIENT_HOST</td>
<td>NVARCHAR(256)</td>
<td>Displays the client host.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP address of the client application.</td>
</tr>
<tr>
<td>CLIENT_PID</td>
<td>BIGINT</td>
<td>Displays the PID of the client process.</td>
</tr>
<tr>
<td>CLIENT_PORT</td>
<td>INTEGER</td>
<td>Displays the port of the client process.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user that is connected to the database.</td>
</tr>
<tr>
<td>STATEMENT_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user who executed the statement.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application user.</td>
</tr>
<tr>
<td>XS_APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the XS application user.</td>
</tr>
<tr>
<td>AUDIT_POLICY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the Audit Policy hit.</td>
</tr>
<tr>
<td>EVENT_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays whether the event was successful or not.</td>
</tr>
<tr>
<td>EVENT_LEVEL</td>
<td>VARCHAR(16)</td>
<td>Displays the severity level of the event.</td>
</tr>
<tr>
<td>EVENT_ACTION</td>
<td>VARCHAR(64)</td>
<td>Displays the action performed by the audit event.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object.</td>
</tr>
<tr>
<td>PRIVILEGE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the granted privilege.</td>
</tr>
</tbody>
</table>

⚠️ Caution

Treat this information with caution. It comes from the application and SAP HANA has no way of verifying its authenticity.
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the granted role.</td>
</tr>
<tr>
<td>ROLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the granted role.</td>
</tr>
<tr>
<td>GRANTEE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the grantee in the GRANT/REVOKE statements.</td>
</tr>
<tr>
<td>GRANTEE</td>
<td>NVARCHAR(256)</td>
<td>Displays the grantee in the GRANT/REVOKE statements.</td>
</tr>
<tr>
<td>GRANTABLE</td>
<td>VARCHAR(16)</td>
<td>Displays whether or not the privilege or role is grantable.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the name of the configuration file that has been changed.</td>
</tr>
<tr>
<td>SECTION</td>
<td>VARCHAR(128)</td>
<td>Displays the configuration that has been changed.</td>
</tr>
<tr>
<td>KEY</td>
<td>NVARCHAR(2000)</td>
<td>Displays the attribute that was changed.</td>
</tr>
<tr>
<td>PREV_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the old value of the attribute.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the new value of the attribute.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the SQL statement that caused the event.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>VARCHAR(5000)</td>
<td>Displays extra information about the event.</td>
</tr>
<tr>
<td>ORIGIN_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the original database name on cross-database queries.</td>
</tr>
<tr>
<td>ORIGIN_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the original user name on cross-database queries.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>(Applies to XSA-events) Displays the time of the event occurrence on the cli-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ent side.</td>
</tr>
<tr>
<td>XSA_MESSAGE_IP</td>
<td>VARCHAR(45)</td>
<td>(Applies to XSA-events) Displays the PI address of the event occurrence.</td>
</tr>
<tr>
<td>XSA_TENANT</td>
<td>VARCHAR(36)</td>
<td>(Applies to XSA-events) Displays the XSA tenant GUID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>XSA_UUID</td>
<td>VARCHAR(256)</td>
<td>(Applies to XSA-events) Displays the unique audit log message ID generated by the audit service.</td>
</tr>
<tr>
<td>XSA_CHANNEL</td>
<td>VARCHAR(16)</td>
<td>(Applies to XSA-events) Displays the communication protocol that was used when the audit event was triggered.</td>
</tr>
<tr>
<td>XSA_ATTACHMENT_ID</td>
<td>NVARCHAR(256)</td>
<td>(Applies to XSA-events) Displays the ID of the attachment that triggered the event.</td>
</tr>
<tr>
<td>XSA_ATTACHMENT_NAME</td>
<td>NVARCHAR(256)</td>
<td>(Applies to XSA-events) Displays the name of the attachment that triggered the event.</td>
</tr>
<tr>
<td>XSA_ORGANIZATION_ID</td>
<td>VARCHAR(36)</td>
<td>(Applies to XSA-events) Displays the application organization GUID.</td>
</tr>
<tr>
<td>XSA_SPACE_ID</td>
<td>VARCHAR(36)</td>
<td>(Applies to XSA-events) Displays the application space GUID.</td>
</tr>
<tr>
<td>XSA_INSTANCE_ID</td>
<td>VARCHAR(36)</td>
<td>(Applies to XSA-events) Displays the GUID of the used auditlog service instance.</td>
</tr>
<tr>
<td>XSA_BINDING_ID</td>
<td>VARCHAR(36)</td>
<td>(Applies to XSA-events) Displays the application binding GUID in regards to the specific auditlog service instance that is being used.</td>
</tr>
<tr>
<td>XSA_OBJECT</td>
<td>NVARCHAR(5000)</td>
<td>(Applies to XSA-events) Displays the object containing the accessed personal data.</td>
</tr>
<tr>
<td>XSA_DATA_SUBJECT</td>
<td>NVARCHAR(5000)</td>
<td>(Applies to XSA-events) Displays the owner of the accessed personal data.</td>
</tr>
</tbody>
</table>

**Additional Information**

Database users with the AUDIT ADMIN, AUDIT OPERATOR, or AUDIT READ system privilege can view information in this system view. For all other database users, this view is empty.
Related Information

AUDIT_ACTIONS System View [page 1500]
AUDIT_LOG System View [page 1501]
AUDIT_POLICIES System View [page 1503]
XSA_AUDIT_LOG System View [page 1727]
GRANT Statement (Access Control) [page 1010]
REVOKE Statement (Access Control) [page 1098]
ALTER SYSTEM CLEAR AUDIT LOG Statement (System Management) [page 517]
Logging and Auditing in XS Advanced
Audit in SAP HANA Cockpit

6.1.16 ANNOTATIONS System View

Provides information about annotations that have been added to SQL objects.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the object.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of object, for example, a table.</td>
</tr>
<tr>
<td>SUBOBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the subobject, if applicable.</td>
</tr>
<tr>
<td>SUBOBJECT_TYPE</td>
<td>VARCHAR(9)</td>
<td>Displays the type of subobject, if applicable, for example, a table column.</td>
</tr>
<tr>
<td>KEY</td>
<td>NVARCHAR(256)</td>
<td>Displays the annotation key.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(256)</td>
<td>Displays the annotation key value.</td>
</tr>
</tbody>
</table>

Related Information

ANNOTATE Statement (Data Definition) [page 686]
6.1.17 ANONYMIZATION_VIEWS System View

Provides information about anonymized views in the SAP HANA database.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the anonymized view.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the anonymized view.</td>
</tr>
<tr>
<td>ALGORITHM</td>
<td>VARCHAR(32)</td>
<td>Displays the algorithm used to anonymize the view: K-ANONYMITY/DIFFERENTIAL PRIVACY.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>NCLOB</td>
<td>Displays the view-level anonymization parameters defined for the view, in JSON format.</td>
</tr>
</tbody>
</table>

Related Information

ANONYMIZATION_VIEW_COLUMNS System View [page 1496]
M_ANONYMIZATION_VIEWS System View [page 1745]
VIEWS System View [page 1702]
CREATE VIEW Statement (Data Definition) [page 904]
ALTER VIEW Statement (Data Definition) [page 667]
REFRESH VIEW Statement (Data Definition) [page 1086]
6.1.18 ANONYMIZATION_VIEW_COLUMNS System View

Provides information about the anonymized columns in SAP HANA database.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the anonymized view.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the anonymized view.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column with anonymization parameters.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the view column.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>NCLOB</td>
<td>Displays the column-level anonymization parameters defined for the view, in JSON format.</td>
</tr>
</tbody>
</table>

**Related Information**

- ANONYMIZATION_VIEWS System View [page 1495]
- M_ANONYMIZATION_VIEWS System View [page 1745]
- VIEWS System View [page 1702]
- CREATE VIEW Statement (Data Definition) [page 904]
- ALTER VIEW Statement (Data Definition) [page 667]
- REFRESH VIEW Statement (Data Definition) [page 1086]
6.1.19 APPLICATION_CONNECTION_HISTORY System View

Provides stored application connection history information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application name.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP address of the client.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CONNECT_DATE</td>
<td>DATE</td>
<td>Displays the established date of the connection.</td>
</tr>
<tr>
<td>CONNECT_COUNT</td>
<td>INTEGER</td>
<td>Displays the count of connections established in a day.</td>
</tr>
</tbody>
</table>

**Related Information**

APPLICATION_ENCRYPTION_KEYS System View [page 1498]
CONNECT Statement (Session Management) [page 721]
M_CONNECTIONS System View [page 1785]
Connections from Database Clients and Web Clients to SAP HANA SQL Connection Details
6.1.20 APPLICATION_ENCRYPTION_KEYS System View

Provides information about encryption keys used by applications.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPONENT</td>
<td>NVARCHAR(20)</td>
<td>Displays the component that the key belongs to.</td>
</tr>
<tr>
<td>SUB_COMPONENT</td>
<td>NVARCHAR(512)</td>
<td>Displays the subcomponent that the key belongs to.</td>
</tr>
<tr>
<td>CREATOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the database user or the internal subcomponent that created the key.</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the key was created.</td>
</tr>
<tr>
<td>IS_CURRENT</td>
<td>VARCHAR(5)</td>
<td>Displays whether the key is used by the subcomponent for newly encrypted data: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

M_ENCRYPTION_OVERVIEW System View [page 1881]
ALTER SYSTEM APPLICATION ENCRYPTION Statement (System Management) [page 512]
ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS Function (Security) [page 187]
ENCRYPTION_ROOT_KEYS_HAS_BACKUP_PASSWORD Function (Security) [page 190]
6.1.21 ASSOCIATIONS System View

Provides information about associations.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the association owner schema name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the association owner table or view name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the association owner object type.</td>
</tr>
<tr>
<td>ASSOCIATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the association name.</td>
</tr>
<tr>
<td>TARGET_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the target schema name that the association is associated with.</td>
</tr>
<tr>
<td>TARGET_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the target table/view name that the association is associated with.</td>
</tr>
<tr>
<td>TARGET_OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the target object type that the association is associated with.</td>
</tr>
<tr>
<td>JOIN_CONDITION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the join condition.</td>
</tr>
<tr>
<td>JOIN_CARDINALITY</td>
<td>VARCHAR(12)</td>
<td>Displays the join cardinality type.</td>
</tr>
<tr>
<td>DEFAULT_FILTER</td>
<td>NVARCHAR(5000)</td>
<td>Displays the default predicate filter for columns.</td>
</tr>
</tbody>
</table>

**Related Information**

CDS_ASSOCIATIONS System View [page 1513]
6.1.22 AUDIT_ACTIONS System View

Provides information about all available audit actions.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION_GROUP</td>
<td>SMALLINT</td>
<td>Displays the ID of the group of audit actions.</td>
</tr>
<tr>
<td>ACTION_GROUP_DESCRIPTION</td>
<td>VARCHAR(64)</td>
<td>Displays the description for the group of audit actions.</td>
</tr>
<tr>
<td>ACTION_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the audit action.</td>
</tr>
<tr>
<td>IS_USER_MANDATORY</td>
<td>VARCHAR(5)</td>
<td>Displays if the audit action needs a user specification: TRUE/FALSE</td>
</tr>
<tr>
<td>IS_OBJECT_MANDATORY</td>
<td>VARCHAR(5)</td>
<td>Displays if the audit action needs an object specification: TRUE/FALSE</td>
</tr>
<tr>
<td>IS_DATABASE_SUPPORTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the audit action is allowed in an audit policy created by an authorized user remotely on SYSTEMDB: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_SCHEMA_SUPPORTED</td>
<td>NVARCHAR(5)</td>
<td>Displays whether this audit action can be specified for specific schemas: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Additional Information

If viewed on a tenant, database users with the CATALOG READ, DATA ADMIN, or AUDIT ADMIN system privilege can view information in this system view. If viewed on SYSTEMDB, users also need the DATABASE AUDIT ADMIN or DATABASE ADMIN system privilege. For all other database users, this view is empty.

Related Information

ALL_AUDIT_LOG System View [page 1490]
AUDIT_LOG System View [page 1501]
AUDIT_POLICIES System View [page 1503]
6.1.23 AUDIT_LOG System View

Provides information about audit records, with the exception of XSA-auditing. You must have the AUDIT ADMIN, AUDIT OPERATOR, or AUDIT READ system privilege to access this view.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the time that the event occurred.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the name of the host where the event occurred.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port number.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the name of the service.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>CLIENT_HOST</td>
<td>NVARCHAR(256)</td>
<td>Displays the IP of the client host.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP of the client application.</td>
</tr>
<tr>
<td>CLIENT_PID</td>
<td>BIGINT</td>
<td>Displays the PID of the client process.</td>
</tr>
<tr>
<td>CLIENT_PORT</td>
<td>INTEGER</td>
<td>Displays the port of the client process.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user that is connected to the database.</td>
</tr>
<tr>
<td>STATEMENT_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user who executed the statement.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application user.</td>
</tr>
<tr>
<td>XS_APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the XS application user.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>AUDIT_POLICY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the Audit Policy hit.</td>
</tr>
<tr>
<td>EVENT_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays whether the event was successful or not.</td>
</tr>
<tr>
<td>EVENT_LEVEL</td>
<td>VARCHAR(16)</td>
<td>Displays the severity level of the event.</td>
</tr>
<tr>
<td>EVENT_ACTION</td>
<td>VARCHAR(64)</td>
<td>Displays the action performed by the audit event.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of schema.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of object.</td>
</tr>
<tr>
<td>PRIVILEGE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the granted privilege.</td>
</tr>
<tr>
<td>ROLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the granted role.</td>
</tr>
<tr>
<td>ROLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the granted role.</td>
</tr>
<tr>
<td>GRANTEE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the grantee in GRANT/REVOKE statements.</td>
</tr>
<tr>
<td>GRANTEE</td>
<td>NVARCHAR(256)</td>
<td>Displays the grantee in GRANT/REVOKE statements.</td>
</tr>
<tr>
<td>GRANTABLE</td>
<td>VARCHAR(16)</td>
<td>Displays whether the privilege/role being granted is grantable or not.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the configuration file that was changed.</td>
</tr>
<tr>
<td>SECTION</td>
<td>VARCHAR(128)</td>
<td>Displays the configuration that was changed.</td>
</tr>
<tr>
<td>KEY</td>
<td>NVARCHAR(2000)</td>
<td>Displays the attribute that was changed.</td>
</tr>
<tr>
<td>PREV_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the old value of the attribute.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the new value of the attribute.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the SQL statement that caused the event.</td>
</tr>
</tbody>
</table>
### Additional Information

Database users with the AUDIT ADMIN, AUDIT OPERATOR, or AUDIT READ system privilege can view information in this system view. For all other database users, this view is empty.

### Related Information

- ALL_AUDIT_LOG System View [page 1490]
- AUDIT_ACTIONS System View [page 1500]
- AUDIT_POLICIES System View [page 1503]
- GRANT Statement (Access Control) [page 1010]
- REVOKE Statement (Access Control) [page 1098]
- ALTER SYSTEM CLEAR AUDIT LOG Statement (System Management) [page 517]
- Audit in SAP HANA Cockpit

### 6.1.24 AUDIT_POLICIES System View

Provides information about audit policies.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIT_POLICY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the audit policy name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AUDIT_POLICY_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the audit policy.</td>
</tr>
<tr>
<td>EVENT_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the status of events to be audited: SUCCESSFUL EVENTS, UNSUCCESSFUL EVENTS, ALL EVENTS.</td>
</tr>
<tr>
<td>EVENT_LEVEL</td>
<td>VARCHAR(16)</td>
<td>Displays the level of events to be audited: EMERGENCY, CRITICAL, ALERT, WARNING, INFO.</td>
</tr>
<tr>
<td>IS_AUDIT_POLICY_ACTIVE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the audit policy is active: TRUE/ FALSE.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(32)</td>
<td>Displays whether an audit policy is valid. An audit policy can be invalid if the referenced user or object does not exist or an invalid combination of action and object is specified (for example, EXECUTE on a table). Possible values are TRUE/ FALSE.</td>
</tr>
<tr>
<td>IS_DATABASE_LOCAL</td>
<td>VARCHAR(5)</td>
<td>Displays whether the policy was created locally on the current database and can be changed from within this database: TRUE/ FALSE.</td>
</tr>
<tr>
<td>EVENT_ACTION</td>
<td>VARCHAR(32)</td>
<td>Displays the action to be audited, such as SELECT or GRANT PRIVILEGE.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user whose actions are to be audited.</td>
</tr>
<tr>
<td>EXCEPT_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user whose actions are not to be audited.</td>
</tr>
<tr>
<td>PRINCIPAL_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the principal whose actions or whose members’ actions are to be audited.</td>
</tr>
<tr>
<td>EXCEPT_PRINCIPAL_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the principal whose actions or whose members’ actions are not to be audited.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PRINCIPAL_TYPE</td>
<td>NVARCHAR(16)</td>
<td>Displays whether the principal or except principal is USER or USERGROUP. When if the value of PRINCIPAL_TYPE is USER, the columns PRINCIPAL_NAME and EXCEPT_PRINCIPAL_NAME display the same values as the columns USER_NAME and EXCEPT_USER_NAME.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of object to be audited, or INVALID if the object type was changed after the audit policy was created.</td>
</tr>
<tr>
<td>OBJECT_SCHEMA</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema of object to be audited.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of object to be audited.</td>
</tr>
<tr>
<td>TRAIL_TYPE</td>
<td>VARCHAR(6)</td>
<td>Displays the name of the audit trail where the audit entry is written.</td>
</tr>
<tr>
<td>RETENTION_PERIOD</td>
<td>INTEGER</td>
<td>Displays the number of days the corresponding audit entries are retained.</td>
</tr>
</tbody>
</table>

**Additional Information**

Database users with the CATALOG READ, DATA ADMIN, or AUDIT ADMIN system privilege can view information in this system view. For all other database users, this view is empty.

**Related Information**

- ALL_AUDIT_LOG System View [page 1490]
- AUDIT_LOG System View [page 1501]
- CREATE AUDIT POLICY Statement (Access Control) [page 722]
- ALTER AUDIT POLICY Statement (Access Control) [page 431]
- DROP AUDIT POLICY Statement (Access Control) [page 946]
- Create an Audit Policy
- Best Practices and Recommendations for Creating Audit Policies
- Auditing Details
- Activate and Configure Auditing
6.1.25 AUTHORIZATION_GRAPH System View

Provides information about authorization dependencies of complex database objects.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT_DEPENDENT_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the object type ID of the root object to show as a graph.</td>
</tr>
<tr>
<td>ROOT_DEPENDENT_OBJECT_ID</td>
<td>BIGINT</td>
<td>Displays the object ID of the root object to show as a graph.</td>
</tr>
<tr>
<td>DEPENDENT_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the object type ID of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_TYPE_ID_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object type name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_SUBTYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the object subtype ID of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_SUBTYPE_ID_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object subtype name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_OBJECT_ID</td>
<td>BIGINT</td>
<td>Displays the object ID of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name the dependent object belongs to.</td>
</tr>
<tr>
<td>DEPENDENT_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_OWNER_OID</td>
<td>BIGINT</td>
<td>Displays the ID of the user owning the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user owning the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_READONLY</td>
<td>INTEGER</td>
<td>Displays the read-only property of the dependent object.</td>
</tr>
<tr>
<td>UNDERLYING_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the object type ID of the underlying object.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UNDERLYING_TYPE_ID_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object type name of the underlying object.</td>
</tr>
<tr>
<td>UNDERLYING_SUBTYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the object subtype ID of the underlying object.</td>
</tr>
<tr>
<td>UNDERLYING_SUBTYPE_ID_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object subtype name of the underlying object.</td>
</tr>
<tr>
<td>UNDERLYING_OBJECT_ID</td>
<td>BIGINT</td>
<td>Displays the object ID of the underlying object.</td>
</tr>
<tr>
<td>UNDERLYING_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name the underlying object belongs to.</td>
</tr>
<tr>
<td>UNDERLYING_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of the underlying object.</td>
</tr>
<tr>
<td>UNDERLYING.Owner_OID</td>
<td>BIGINT</td>
<td>Displays the ID of the user owning the underlying object.</td>
</tr>
<tr>
<td>UNDERLYING.Owner_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user owning the underlying object.</td>
</tr>
<tr>
<td>UNDERLYING_READONLY</td>
<td>INTEGER</td>
<td>Displays the read-only property of the underlying object.</td>
</tr>
<tr>
<td>DEPENDENCY_USER_OID</td>
<td>BIGINT</td>
<td>Displays the ID of the user required to have a certain privilege on the underlying object in order to validate the dependent object.</td>
</tr>
<tr>
<td>DEPENDENCY_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user required to have a certain privilege on the underlying object in order to validate the dependent object.</td>
</tr>
<tr>
<td>DEPENDENCY_TYPE</td>
<td>SMALLINT</td>
<td>Displays the type of dependency that determines the validation semantics for dependencies sharing the same dependent objects.</td>
</tr>
<tr>
<td>PRIVILEGE_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the privilege the dependency user is required to have on the underlying object.</td>
</tr>
<tr>
<td>PRIVILEGE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the privilege the dependency user is required to have on the underlying object.</td>
</tr>
</tbody>
</table>
### Additional Information

This view requires an equal to predicate on ROOT_DEPENDENT_TYPE_ID and ROOT_DEPENDENT_OBJECT_ID.

### Related Information

**AUTHORIZATION_TYPES System View** [page 1509]

#### 6.1.26 AUTHORIZATION_OBJECTS System View

Provides information regarding authorization objects.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the authorization object.</td>
</tr>
</tbody>
</table>
### Related Information

- AUTHORIZATION_GRAPH System View [page 1506]
- AUTHORIZATION_TYPES System View [page 1509]
- Classification of Authorization Dependencies Between Objects
- Object Privileges
- Authorizations Needed for Monitoring and Administration
- Create and Authorize a User

### 6.1.27 AUTHORIZATION_TYPES System View

Provides information about object types and subtypes used by authorization object IDs.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the authorization object type.</td>
</tr>
<tr>
<td>TYPE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the authorization object type.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SUBTYPE_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the authorization object subtype.</td>
</tr>
<tr>
<td>SUBTYPE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the authorization object subtype.</td>
</tr>
</tbody>
</table>

**Related Information**

- AUTHORIZATION_GRAPH System View [page 1506]
- AUTHORIZATION_OBJECTS System View [page 1508]

**6.1.28 CDS_ANNOTATION.Assignments System View**

Provides CDS annotation assignments.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>ARTIFACT_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the artifact name.</td>
</tr>
<tr>
<td>COMPONENT_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the component name.</td>
</tr>
<tr>
<td>ANNOTATION_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the annotation name.</td>
</tr>
<tr>
<td>EXTENSION_PACKAGE_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the extension package name.</td>
</tr>
<tr>
<td>FORMAT</td>
<td>NVARCHAR(127)</td>
<td>Displays the format of the annotation value.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NCLOB(2147483647)</td>
<td>Displays the value.</td>
</tr>
<tr>
<td>VALIDATION_RESULT</td>
<td>NVARCHAR(127)</td>
<td>Displays the validation result of the annotation value.</td>
</tr>
<tr>
<td>DEFINITION_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the annotation definition.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>DEFINITION_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the name of the annotation definition.</td>
</tr>
</tbody>
</table>

**Related Information**

- CDS Annotations
- User-Defined CDS Annotations
- CDS Annotation Usage Examples
- ANNOTATIONS System View [page 1494]
- CDS_ANNOTATION_VALUES System View [page 1511]
- ANNOTATE Statement (Data Definition) [page 686]
- CDS_ARTIFACT_NAMES System View [page 1512]
- CDS_ASSOCIATIONS System View [page 1513]
- CDS_ENTITIES System View [page 1514]
- CDS_VIEWS System View [page 1515]

**6.1.29 CDS_ANNOTATION_VALUES System View**

Provides CDS annotation values.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>ARTIFACT_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the artifact name.</td>
</tr>
<tr>
<td>ELEMENT_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the element name.</td>
</tr>
<tr>
<td>ANNOTATION_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the annotation schema name.</td>
</tr>
<tr>
<td>ANNOTATION_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the annotation name.</td>
</tr>
<tr>
<td>EXTENSION_PACKAGE_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the extension package name.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NCLOB</td>
<td>Displays the value of the annotation.</td>
</tr>
</tbody>
</table>
6.1.30 CDS_ARTIFACT_NAMES System View

Provides CDS artifact names.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>ARTIFACT_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the artifact name.</td>
</tr>
<tr>
<td>ARTIFACT_KIND</td>
<td>VARCHAR(32)</td>
<td>Displays the kind of artifact: ANNOTATION, ARRAY_TYPE, ASSOCIATION, CONTEXT,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DERIVED_TYPE, ENTITY, ENUM, STRUCTURED_TYPE, or VIEW.</td>
</tr>
</tbody>
</table>

Related Information

CDS_ANNOTATION_ASSIGNMENTS System View [page 1510]
CDS_ANNOTATION_VALUES System View [page 1511]
CDS_ARTIFACT_NAMES System View [page 1512]
CDS_ASSOCIATIONS System View [page 1513]
CDS_ENTITIES System View [page 1514]
CDS_VIEWS System View [page 1515]
6.1.31  CDS_ASSOCIATIONS System View

Provides definitions for CDS associations.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>ASSOCIATION_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the association name.</td>
</tr>
<tr>
<td>ASSOCIATION_KIND</td>
<td>VARCHAR(32)</td>
<td>Displays the kind of association: FOREIGN KEY EXPLICIT, FOREIGN KEY IMPLICIT, or UNMANAGED.</td>
</tr>
<tr>
<td>EXTENSION_PACKAGE_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the extension package name.</td>
</tr>
<tr>
<td>TARGET_ARTIFACT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the associated CDS artifact.</td>
</tr>
<tr>
<td>TARGET_ARTIFACT_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the name of the associated CDS artifact.</td>
</tr>
<tr>
<td>TARGET_CARDINALITY_MIN</td>
<td>INTEGER</td>
<td>Displays the minimum cardinality of the association target. The default is 0. -1 represents unlimited.</td>
</tr>
<tr>
<td>TARGET_CARDINALITY_MAX</td>
<td>INTEGER</td>
<td>Displays the maximum cardinality of the association target. The default is 1.</td>
</tr>
<tr>
<td>JOIN_CONDITION</td>
<td>NCLOB</td>
<td>Displays the join condition for unmanaged associations (ASSOCIATION_KIND = UNMANAGED).</td>
</tr>
</tbody>
</table>

Additional Information

Association definitions are exposed as part of the CDS_ARTIFACT_DEFINITION result set, represented by the ARTIFACT_KIND association. Foreign key elements are also available from the ASSOCIATION_ELEMENT, which distinguishes them from regular elements.
Related Information

Create an Association in CDS
CDS Associations
CDS Association Syntax Options
CDS_ANNOTATION_ASSIGNMENTS System View [page 1510]
CDS_ANNOTATION_VALUES System View [page 1511]
CDS_ARTIFACT_NAMES System View [page 1512]
CDS_ENTITIES System View [page 1514]
CDS_VIEWS System View [page 1515]

6.1.32 CDS_ENTITIES System View

Provides information about CDS entities.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>ENTITY_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the entity name.</td>
</tr>
<tr>
<td>EXTENSION_PACKAGE_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the extension package name.</td>
</tr>
<tr>
<td>SERIES_KIND</td>
<td>VARCHAR(32)</td>
<td>Displays the kind of series: NO_SERIES, NOT_EQUIDISTANT, EQUIDISTANT, or EQUIDISTANT_PIECEWISE.</td>
</tr>
</tbody>
</table>

Related Information

Create an Entity in CDS
CDS Entities
Entity Element Modifiers
CDS Entity Syntax Options
CDS_ANNOTATION_ASSIGNMENTS System View [page 1510]
CDS_ANNOTATION_VALUES System View [page 1511]
CDS_ASSOCIATIONS System View [page 1513]
CDS_ARTIFACT_NAMES System View [page 1512]
CDS_ENTITIES System View [page 1514]
6.1.33 CDS_VIEWS System View

Provides definitions for CDS views.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the view name.</td>
</tr>
<tr>
<td>EXTENSION_PACKAGE_NAME</td>
<td>NVARCHAR(127)</td>
<td>Displays the extension package name.</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NCLOB</td>
<td>Displays the original definition of the view, similar to how it is found in</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the CDS resource file, but with the relative names resolved to absolute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>names, and constant expressions resolved to concrete values.</td>
</tr>
</tbody>
</table>

**Related Information**

Create a View in CDS
CDS Views
CDS View Syntax Options
CDS_ANNOTATION_ASSIGNMENTS System View [page 1510]
CDS_ANNOTATION_VALUES System View [page 1511]
CDS_ARTIFACT_NAMES System View [page 1512]
CDS_ASSOCIATIONS System View [page 1513]
CDS_ENTITIES System View [page 1514]
### 6.1.34 CERTIFICATES System View

Provides information about certificates usable in PSEs.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CERTIFICATE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the unique name of the certificate.</td>
</tr>
<tr>
<td>CERTIFICATE_ID</td>
<td>BIGINT</td>
<td>Displays the unique identification of the certificate.</td>
</tr>
<tr>
<td>SUBJECT_COMMON_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the distinguished name of the X.509 certificate subject.</td>
</tr>
<tr>
<td>ISSUER_COMMON_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the common name of the X.509 certificate issuer.</td>
</tr>
<tr>
<td>VALID_FROM</td>
<td>TIMESTAMP</td>
<td>Displays the start time of certificate’s validity.</td>
</tr>
<tr>
<td>VALID_UNTIL</td>
<td>TIMESTAMP</td>
<td>Displays the end time of certificate’s validity.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>NVARCHAR(1024)</td>
<td>Displays the description for the certificate.</td>
</tr>
<tr>
<td>SUBJECT_DISTINGUISHED_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the distinguished name of the X.509 certificate subject.</td>
</tr>
<tr>
<td>ISSUER_DISTINGUISHED_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the distinguished name of the X.509 certificate issuer.</td>
</tr>
<tr>
<td>BASIC_CONSTRAINTS</td>
<td>NVARCHAR(32)</td>
<td>Displays the basic constraints.</td>
</tr>
<tr>
<td>SUBJECT_ALT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the comma-separated list of alternative names of the X.509 certificate subject.</td>
</tr>
<tr>
<td>PUBLIC_KEY_ALGORITHM</td>
<td>NVARCHAR(256)</td>
<td>Displays the public key algorithm.</td>
</tr>
<tr>
<td>PUBLIC_KEY_LENGTH</td>
<td>SMALLINT</td>
<td>Displays the public key length.</td>
</tr>
<tr>
<td>KEY_USAGE</td>
<td>NVARCHAR(256)</td>
<td>Displays the comma-separated list of key usage strings as specified in RFC 5280.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SIGNATURE_ALGORITHM</td>
<td>NVARCHAR(256)</td>
<td>Displays the signature algorithm.</td>
</tr>
<tr>
<td>VERSION</td>
<td>TINYINT</td>
<td>Displays the version of the certificate format.</td>
</tr>
<tr>
<td>SERIAL_NUMBER</td>
<td>NVARCHAR(64)</td>
<td>Displays the serial number as assigned from the certificate issuer.</td>
</tr>
<tr>
<td>FINGERPRINT</td>
<td>NVARCHAR(256)</td>
<td>Displays the hash of the entire certificate. It is used as a unique identifier in the certificate store.</td>
</tr>
<tr>
<td>CERTIFICATE</td>
<td>NCLOB</td>
<td>Displays the certificate as given during SQL.</td>
</tr>
</tbody>
</table>

**Related Information**

- CREATE CERTIFICATE Statement (System Management) [page 739]
- DROP CERTIFICATE Statement (System Management) [page 948]
- PSE_CERTIFICATES System View [page 1620]

**6.1.35 CLIENTSIDE_ENCRYPTION_COLUMN_KEYS System View**

Provides column encryption key information.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema.</td>
</tr>
<tr>
<td>COLUMN_KEY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column encryption key.</td>
</tr>
<tr>
<td>COLUMN_KEY_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the UUID of the key.</td>
</tr>
<tr>
<td>ENCRYPTED_WITH_KEYPAIR_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the UUID of the public key that this key is encrypted with.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
ENCRYPTED_COLUMN_KEYVALUE | VARBINARY(1024) | Displays the value of the key encrypted with the public key referenced by the ENCRYPTED_WITH_KEYPAIR_ID column.
ENCRYPTION_ALGORITHM | VARCHAR(16) | Displays the algorithm that this key uses to encrypt data. Possible values: AES-256-CBC and ARIA-256-CBC.
CREATE_TIME | TIMESTAMP | Displays the time that the key was created.
CREATOR | NVARCHAR(256) | Displays the name of the user who created this key.
CREATED_FOR_COLUMN_KEY_ADMIN | VARCHAR(5) | Displays whether created for administrator key copy: TRUE/FALSE.
COLUMN_KEY_VERSION | INTEGER | Displays the version of the column encryption key.
ALTER_TABLE_IN_PROGRESS | VARCHAR(5) | Displays whether an ALTER TABLE operation is in process using the CEK: TRUE/FALSE.

### Additional Information
You must have the CLIENTSIDE ENCRYPTION COLUMN KEY ADMIN privilege to access this view.

### Related Information
- CLIENTSIDE_ENCRYPTION_KEYPAIRS System View [page 1519]
- CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 741]
- ALTER CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 435]
- DROP CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 949]
- CREATE CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 743]
- DROP CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 950]
6.1.36 CLIENTSIDE_ENCRYPTION_KEYPAIRS System View

Provides information about client public keys in the SAP HANA database.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEYPAIR_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the client public key.</td>
</tr>
<tr>
<td>KEYPAIR_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the UUID key identifier.</td>
</tr>
<tr>
<td>ENCODED_PUBLIC_KEYVALUE</td>
<td>NCLOB</td>
<td>Displays the PEM-encoded x509 value of the key.</td>
</tr>
<tr>
<td>ENCRYPTION_ALGORITHM</td>
<td>VARCHAR(16)</td>
<td>Displays the algorithm the key uses to encrypt column keys. Possible values are: RSA-OAEP-2048.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time the key was created.</td>
</tr>
<tr>
<td>CREATOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user who created the key.</td>
</tr>
<tr>
<td>KEYPAIR_VERSION</td>
<td>INTEGER</td>
<td>Displays the version of the key.</td>
</tr>
</tbody>
</table>

Related Information

CLIENTSIDE_ENCRYPTION_COLUMN.Keys System View [page 1517]
CREATE CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 741]
ALTER CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 435]
DROP CLIENTSIDE ENCRYPTION COLUMN KEY Statement (Encryption) [page 949]
CREATE CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 743]
DROP CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 950]
ALTER CLIENTSIDE ENCRYPTION KEYPAIR Statement (Encryption) [page 437]
6.1.37 COLLATIONS System View

Provides the list of collations that can be specified in an ORDER BY clause.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLATION_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the supported collation names.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(64)</td>
<td>Displays the language and region description for the collation.</td>
</tr>
</tbody>
</table>

Additional information

The presence of CI or AI at the end of a collation name indicates that the collation is case-insensitive or accent-insensitive, respectively (for example, FRENCH_AI).

Related Information

AUTO_CORR Function (Aggregate) [page 108]
CROSS_CORR Function (Aggregate) [page 157]
DFT Function (Aggregate) [page 185]
FIRST_VALUE Function (Aggregate) [page 195]
LAST_VALUE Function (Aggregate) [page 238]
NTH_VALUE Function (Aggregate) [page 279]
SELECT Statement (Data Manipulation) [page 1104]
STRING_AGG Function (Aggregate) [page 351]
Window Functions and the Window Specification [page 424]

6.1.38 COLUMNS System View - Deprecated

This view is deprecated. Use either the TABLE_COLUMNS or VIEW_COLUMNS system view instead.
## 6.1.39 CONFIGURATION_PARAMETER_PROPERTIES System View

Displays metadata and properties of the public configuration parameters for SAP HANA.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECTION</td>
<td>VARCHAR(128)</td>
<td>Displays the section of the parameter to configure.</td>
</tr>
<tr>
<td>KEY</td>
<td>VARCHAR(128)</td>
<td>Displays the parameter key.</td>
</tr>
<tr>
<td>HAS_KEY_INDEX</td>
<td>VARCHAR(9)</td>
<td>Displays whether or not the parameter is indexed: NO, OPTIONAL, or MANDATORY.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>NVARCHAR(5000)</td>
<td>Displays a description of the parameter.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the data type of the parameter.</td>
</tr>
<tr>
<td>VALUE_LIST_SEPARATOR</td>
<td>VARCHAR(1)</td>
<td>Displays the element separator character for value lists. Returns NULL for simple values.</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the default value of the parameter.</td>
</tr>
<tr>
<td>RESTART_REQUIRED</td>
<td>VARCHAR(8)</td>
<td>Displays whether or not a restart is required. The values are: TRUE, FALSE, or CUSTOM. CUSTOM indicates a custom function or callback to decide whether a restart is required or not depending on the new value to be set.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INIFILE_NAMES</td>
<td>VARCHAR(32)</td>
<td>Displays the .ini files that can contain the parameter value, for example: service.ini, &lt;service&gt;.ini, global.ini</td>
</tr>
<tr>
<td>UNIT</td>
<td>VARCHAR(16)</td>
<td>Displays the unit that the value is measured in, for example, megabytes, kilobytes, seconds, and so on.</td>
</tr>
<tr>
<td>VALUE_RESTRICTIONS</td>
<td>NVARCHAR(256)</td>
<td>Displays the restrictions for the supported values.</td>
</tr>
<tr>
<td>CUSTOM_RESTRICTIONS</td>
<td>NVARCHAR(1024)</td>
<td>Displays the description of the custom restriction, if one is defined.</td>
</tr>
<tr>
<td>LAYER_RESTRICTIONS</td>
<td>VARCHAR(8)</td>
<td>Displays the layer name if a parameter can only be set on specific layers: SYSTEM, HOST, or READONLY.</td>
</tr>
<tr>
<td>IS_READ_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays if the parameter is set to read only.</td>
</tr>
<tr>
<td>IS_SYSTEM_MANAGED</td>
<td>VARCHAR(5)</td>
<td>Displays if a parameter is managed internally by the system.</td>
</tr>
<tr>
<td>IS_TEMPLATE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the configuration parameter templates or regular parameters are used: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

- SAP HANA Configuration Parameter Reference
- Configuration Parameters
- Backup Configuration Parameters
- Database-Specific Configuration Parameters
- SAP HANA System Replication Configuration Parameters
- System Properties
6.1.40  CONSTRAINTS System View

Provides information about defined constraints for tables.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>POSITION</td>
<td>SMALLINT</td>
<td>Displays the column position in this constraint.</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the constraint name.</td>
</tr>
<tr>
<td>IS_PRIMARY_KEY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the constraint is a primary key constraint: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_UNIQUE_KEY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the constraint is a unique constraint: TRUE/FALSE.</td>
</tr>
<tr>
<td>CHECK_CONDITION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the check condition of the check constraint.</td>
</tr>
</tbody>
</table>

Related Information

REFERENTIAL_CONSTRAINTS System View [page 1625]
CS_JOIN_CONSTRAINTS System View [page 1530]
M_TEMPORARY_JOIN_CONSTRAINTS System View [page 2233]
System Limitations [page 1208]
6.1.41 CONVERT_INTERNAL_TYPE_INTO_NAME System View

For internal use only.

6.1.42 CREDENTIALS System View

Provides information about credentials for users and components.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user the credential belongs to.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the component the credential is valid for.</td>
</tr>
<tr>
<td>PURPOSE</td>
<td>NVARCHAR(256)</td>
<td>Displays the purpose, for example remote machine, that the credential is valid for.</td>
</tr>
<tr>
<td>CREDENTIAL_TYPE</td>
<td>NVARCHAR(64)</td>
<td>Displays the type of the credential, for example, password or X509.</td>
</tr>
</tbody>
</table>

Related Information

CREATE CREDENTIAL Statement (Access Control) [page 744]
ALTER CREDENTIAL Statement (Access Control) [page 439]
DROP CREDENTIAL Statement (Access Control) [page 953]
Managing Credentials
Managing Secondary Credentials
6.1.43 CS_ALL_COLUMNS System View

Provides information from all columns of column tables, including internal ones.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the available table columns.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the parameter.</td>
</tr>
<tr>
<td>GENERATION_TYPE</td>
<td>VARCHAR(22)</td>
<td>Displays the generation type of the column. ALWAYS AS appears if the column is a generation column. ALWAYS AS IDENTITY or BY DEFAULT AS IDENTITY appears if the column is an identity column whose values are always generated or generated by default. If the column is neither generation nor identify, NULL appears.</td>
</tr>
<tr>
<td>GENERATED_ALWAYS_AS</td>
<td>NVARCHAR(1000)</td>
<td>Displays the expression of the column created by GENERATED ALWAYS AS.</td>
</tr>
<tr>
<td>INTERNAL_ATTRIBUTE_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the attribute type for internal columns, otherwise NULL.</td>
</tr>
<tr>
<td>INTERNAL_COLUMN_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the internal column.</td>
</tr>
</tbody>
</table>

Related Information

M_CS_ALL_COLUMNS System View [page 1807]
M_CS_ALL_COLUMN_STATISTICS System View [page 1813]
M_CS_COLUMNS System View [page 1814]
M_CS_COLUMNS_PERSISTENCE System View [page 1818]
CS_BO_VIEWS System View [page 1526]
CS_CONCAT_COLUMNS System View [page 1527]
CS_FREESTYLE_COLUMNS System View [page 1528]
6.1.44 CS_BO_VIEWS System View

Provides information about business object views for column store join views.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join view name.</td>
</tr>
<tr>
<td>BO_VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the business object view name.</td>
</tr>
<tr>
<td>ANCHOR_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the anchor schema name.</td>
</tr>
<tr>
<td>ANCHOR_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the anchor table name.</td>
</tr>
</tbody>
</table>

Related Information

CS_ALL_COLUMNS System View [page 1525]
CS_BO_VIEWS System View [page 1526]
CS_CONCAT_COLUMNS System View [page 1527]
CS_FREESTYLE_COLUMNS System View [page 1528]
CS_JOIN_CONDITIONS System View [page 1529]
CS_JOIN_CONSTRAINTS System View [page 1530]
CS_JOIN_PATHS System View [page 1531]
CS_JOIN_TABLES System View [page 1532]
CS_KEY FIGURES System View [page 1533]
CS_VIEW_COLUMNS System View [page 1535]
CS_VIEW_PARAMETERS System View [page 1536]
6.1.45 CS_CONCAT_COLUMNS System View

Provides information on concat columns in the database.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>CONCAT_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the concat column name.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the position of the column.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
</tbody>
</table>

Related Information

CS_ALL_COLUMNS System View [page 1525]
M_CS_ALL_COLUMNS System View [page 1807]
M_CS_ALL_COLUMN_STATISTICS System View [page 1813]
M_CS_COLUMNS System View [page 1814]
M_CS_COLUMNS_PERSISTENCE System View [page 1818]
CS_VO_VIEWS System View [page 1526]
CS_FREESTYLE_COLUMNS System View [page 1528]
CS_JOIN_CONDITIONS System View [page 1529]
CS_JOIN_CONSTRAINTS System View [page 1530]
CS_JOIN_PATHS System View [page 1531]
CS_JOIN_TABLES System View [page 1532]
CS_KEY FIGURES System View [page 1533]
CS_VIEW_COLUMNS System View [page 1535]
CS_VIEW_PARAMETERS System View [page 1536]
Memory Management in the Column Store
CONCAT Function (String) [page 129]
Concatenate Two Arrays

6.1.46 CS_FREESTYLE_COLUMNS System View

Provides freestyle search columns for column store join views.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join view name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the freestyle column name.</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>REAL</td>
<td>Displays the freestyle weight.</td>
</tr>
<tr>
<td>IS_ALPHANUM_SEARCH</td>
<td>VARCHAR(5)</td>
<td>Displays whether the freestyle search type is alphanumeric: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

CS_ALL_COLUMNS System View [page 1525]
M_CS_ALL_COLUMNS System View [page 1807]
M_CS_ALL_COLUMN_STATISTICS System View [page 1813]
M_CS_COLUMNS System View [page 1814]
M_CS_COLUMNS_PERSISTENCE System View [page 1818]
CS_BO_VIEWS System View [page 1526]
CS_CONCAT_COLUMNS System View [page 1527]
CS_JOIN_CONDITIONS System View [page 1529]
CS_JOIN_CONSTRAINTS System View [page 1530]
CS_KEY.Paths System View [page 1531]
CS_JOIN_TABLES System View [page 1532]
CS_KEY_FIGURES System View [page 1533]
CS_VIEW_COLUMNS System View [page 1535]
CS_VIEW_PARAMETERS System View [page 1536]
Memory Management in the Column Store
6.1.47 CS_JOIN_CONDITIONS System View

Provides join conditions for column store join views.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join view name.</td>
</tr>
<tr>
<td>JOIN_CONDITION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join condition name.</td>
</tr>
<tr>
<td>JOIN_ORDER</td>
<td>BIGINT</td>
<td>Displays the join order number.</td>
</tr>
<tr>
<td>TABLE_SCHEMA_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of column 1.</td>
</tr>
<tr>
<td>TABLE_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of COLUMN_NAME1.</td>
</tr>
<tr>
<td>COLUMN_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of COLUMN_NAME1.</td>
</tr>
<tr>
<td>ALIAS_NUMBER1</td>
<td>INTEGER</td>
<td>Displays the alias number of TABLE_NAME1.</td>
</tr>
<tr>
<td>TABLE_SCHEMA_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of TABLE_NAME2.</td>
</tr>
<tr>
<td>TABLE_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of COLUMN_NAME2.</td>
</tr>
<tr>
<td>COLUMN_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of COLUMN_NAME2.</td>
</tr>
<tr>
<td>ALIAS_NUMBER2</td>
<td>INTEGER</td>
<td>Displays the alias number of TABLE_NAME2.</td>
</tr>
<tr>
<td>CONSTRAINTS</td>
<td>NVARCHAR(256)</td>
<td>Displays the join constraint name.</td>
</tr>
<tr>
<td>JOIN_TYPE</td>
<td>VARCHAR(5)</td>
<td>Displays the join type: INNER, LEFT, RIGHT, FULL.</td>
</tr>
<tr>
<td>CARDINALITY</td>
<td>VARCHAR(3)</td>
<td>Displays the join cardinality: 1:1, 1:n, n:1, n:n.</td>
</tr>
</tbody>
</table>
Related Information

M_TEMPORARY_JOIN_CONDITIONS System View [page 2232]
CS.Join_CONSTRAINTS System View [page 1530]
CS_JOIN_PATHS System View [page 1531]
CS JOIN TABLES System View [page 1532]
CS_ALL_COLUMNS System View [page 1525]
M_CS_ALL_COLUMNS System View [page 1807]
M_CS_ALL_COLUMN_STATISTICS System View [page 1813]
M_CS_COLUMNS System View [page 1814]
M_CS_COLUMNS_PERSISTENCE System View [page 1818]
CS_BO_VIEWS System View [page 1526]
CS_CONCAT_COLUMNS System View [page 1527]
CS_KEY_FIGURES System View [page 1533]
CS_VIEW_COLUMNS System View [page 1535]
CS_VIEW_PARAMETERS System View [page 1536]
Memory Management in the Column Store
SELECT Statement (Data Manipulation) [page 1104]

6.1.48 CS_JOIN_CONSTRAINTS System View

Provides join constraints for column store join views.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join view name.</td>
</tr>
<tr>
<td>JOIN_CONSTRAINT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join constraint name.</td>
</tr>
<tr>
<td>CONSTRAINT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of constraint owner.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of constraint owner.</td>
</tr>
<tr>
<td>TABLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of the column.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the operator.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the value.</td>
</tr>
</tbody>
</table>

**Related Information**

- CS_JOIN_CONDITIONS System View [page 1529]
- CS_JOIN_PATHS System View [page 1531]
- CS_JOIN_TABLES System View [page 1532]
- M_TEMPORARY_JOIN_CONSTRAINTS System View [page 2233]
- CONSTRAINTS System View [page 1523]
- REFERENTIAL_CONSTRAINTS System View [page 1625]
- System Limitations [page 1208]
- CS_ALL_COLUMNS System View [page 1525]
- CS_BO_VIEWS System View [page 1526]
- CS_CONCAT_COLUMNS System View [page 1527]
- CS_FREESTYLE_COLUMNS System View [page 1528]
- CS_KEY FIGURES System View [page 1533]
- CS_VIEW_COLUMNS System View [page 1535]
- CS_VIEW_PARAMETERS System View [page 1536]

Memory Management in the Column Store

### 6.1.49 CS_JOIN_PATHS System View

Provides join paths for column store join views.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join view name.</td>
</tr>
<tr>
<td>JOIN_PATH_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join path name.</td>
</tr>
</tbody>
</table>
### Join Conditions and Constraints

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOIN_CONDITIONS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the comma separated list of join conditions.</td>
</tr>
<tr>
<td>JOIN_CONSTRAINTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the comma separated list of join constraints.</td>
</tr>
</tbody>
</table>

### Related Information

- CS_JOIN_CONSTRAINTS System View [page 1530]
- CS_JOIN_CONDITIONS System View [page 1529]
- CS_JOIN_TABLES System View [page 1532]
- M_TEMPORARY_JOIN_CONSTRAINTS System View [page 2233]
- CS_ALL_COLUMNS System View [page 1525]
- CS_BO_VIEWS System View [page 1526]
- CS_CONCAT_COLUMNS System View [page 1527]
- CS_FREESTYLE_COLUMNS System View [page 1528]
- CS_KEY FIGURES System View [page 1533]
- CS_VIEW_COLUMNS System View [page 1535]
- CS_VIEW_PARAMETERS System View [page 1536]

### Memory Management in the Column Store

### 6.1.50 CS_JOIN_TABLES System View

Provides information about the physical tables referred to by column store join views.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join view name.</td>
</tr>
<tr>
<td>TABLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the physical table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the physical table name.</td>
</tr>
</tbody>
</table>
### Related Information

- **CS_JOIN_PATHS** System View [page 1531]
- **CS_JOIN_CONSTRAINTS** System View [page 1530]
- **CS_JOIN_CONDITIONS** System View [page 1529]
- **M_TEMPORARY_JOIN_CONSTRAINTS** System View [page 2233]
- **CS_ALL_COLUMNS** System View [page 1525]
- **CS_BO_VIEWS** System View [page 1526]
- **CS_CONCAT_COLUMNS** System View [page 1527]
- **CS_FREESTYLE_COLUMNS** System View [page 1528]
- **CS_KEY_FIGURES** System View [page 1533]
- **CS_VIEW_COLUMNS** System View [page 1535]
- **CS_VIEW_PARAMETERS** System View [page 1536]

**Memory Management in the Column Store**

### 6.1.51 CS_KEY_FIGURES System View

Provides information about the key figures defined for column store join views.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join view name.</td>
</tr>
<tr>
<td>KEY FIGURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the key figure name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DEFAULT_AGGREGATION_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the aggregation type: SUM, COUNT, MIN, MAX, and so on.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the description.</td>
</tr>
<tr>
<td>UNIT_CONVERSION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the unit conversion.</td>
</tr>
<tr>
<td>TABLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Table name of column.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>EXPRESSION_FLAGS</td>
<td>VARCHAR(32)</td>
<td>Displays the expression flags.</td>
</tr>
<tr>
<td>EXPRESSION</td>
<td>NVARCHAR(256)</td>
<td>Displays the expression.</td>
</tr>
</tbody>
</table>

**Related Information**

Memory Management in the Column Store
- CS_VIEW_PARAMETERS System View [page 1536]
- CS_VIEW_COLUMNS System View [page 1535]
- CS_FREESTYLE_COLUMNS System View [page 1528]
- CS_JOIN_TABLES System View [page 1532]
- CS_JOIN_PATHS System View [page 1531]
- CS_JOIN_CONSTRAINTS System View [page 1530]
- CS_JOIN_CONDITIONS System View [page 1529]
- CS_CONCAT_COLUMNS System View [page 1527]
- CS_BO_VIEWS System View [page 1526]
- M_CS_COLUMNS_PERSISTENCE System View [page 1818]
- M_CS_COLUMNS System View [page 1814]
- M_CS_ALL_COLUMN_STATISTICS System View [page 1813]
- M_CS_ALL_COLUMNS System View [page 1807]
- CS_ALL_COLUMNS System View [page 1525]
- CE_OLAP_VIEW
6.1.52 CS_VIEW_COLUMNS System View

Provides information about the columns defined for column store join views.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join view name.</td>
</tr>
<tr>
<td>VIEW_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view column name.</td>
</tr>
<tr>
<td>TABLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of column.</td>
</tr>
<tr>
<td>ALIAS_NUMBER</td>
<td>INTEGER</td>
<td>Displays the alias number.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>JOIN_PATH</td>
<td>NVARCHAR(256)</td>
<td>Displays the join path name.</td>
</tr>
<tr>
<td>CS_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the column store data type.</td>
</tr>
<tr>
<td>EXPRESSION_FLAGS</td>
<td>VARCHAR(32)</td>
<td>Displays the expression flags.</td>
</tr>
<tr>
<td>EXPRESSION</td>
<td>NVARCHAR(256)</td>
<td>Displays the expression.</td>
</tr>
<tr>
<td>IS_TEXT_SEARCHABLE.</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is text searchable: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_FUZZY_SEARCHABLE.</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is fuzzy searchable: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

- CS_ALL_COLUMNS System View [page 1525]
- M_CS_ALL_COLUMNS System View [page 1807]
- M_CS_ALL_COLUMN_STATISTICS System View [page 1813]
- M_CS_COLUMNS System View [page 1814]
- M_CS_COLUMNS_PERSISTENCE System View [page 1818]
- CS_BO_VIEWS System View [page 1526]
- CS_CONCAT_COLUMNS System View [page 1527]
6.1.53 CS_VIEW_PARAMETERS System View

Provides a list of parameters of the objects in the SAP HANA database. Only calculation views are considered. The parameters of a view are parsed from the definition of the underlying scenario.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema in which the object is deployed.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object. Only the calculation views are listed. The calculation views are not shown if the underlying calculation scenario has no parameter.</td>
</tr>
</tbody>
</table>
| PARAMETER_NAME       | NVARCHAR(256) | Displays the name of the parameter. If a parameter is referenced by different views, then the parameters are listed for each view separately. Normally, the parameters in a calculation views begins and ends with "$$" character. For example: $$VAR$$.
<p>| IS_MANDATORY         | VARCHAR(5)    | Displays whether a parameter is mandatory or not: TRUE/FALSE. If TRUE, the value of the parameter should be provided either by setting the default value or through the SQL query. If FALSE, setting the value of the parameter is optional. |</p>
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_VALUE</td>
<td>NVARCHAR(256)</td>
<td>Displays the default value of the parameter. This column is set to an empty string when the parameter has no default value.</td>
</tr>
<tr>
<td>EVALUATED_DEFAULT_VALUE</td>
<td>NVARCHAR(256)</td>
<td>Displays the result of the evaluation of the expression in the DEFAULT_VALUE column.</td>
</tr>
<tr>
<td>IS_DEFAULT_VALUE_VOLATILE</td>
<td>BOOLEAN</td>
<td>Displays whether or not the default value is volatile: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

### Additional Information

Executing a SELECT statement on this monitoring view can be very expensive since the scenarios that have not been cached are fetched from the repository, parsed, and their parameters are extracted. However, providing SCHEMA_NAME and OBJECT_NAME columns can significantly decrease the amount of query time.

### Related Information

- VIEW_PARAMETERS System View [page 1709]
- CS_ALL_COLUMNS System View [page 1525]
- CS_BO_VIEWS System View [page 1526]
- CS_CONCAT_COLUMNS System View [page 1527]
- CS_FREESTYLE_COLUMNS System View [page 1528]
- CS_JOIN_CONDITIONS System View [page 1529]
- CS_JOIN_CONSTRAINTS System View [page 1530]
- CS_JOIN_PATHS System View [page 1531]
- CS_JOIN_TABLES System View [page 1532]
- CS_KEY_FIGURES System View [page 1533]
- CS_VIEW_COLUMNS System View [page 1535]
- Memory Management in the Column Store
- SELECT Statement (Data Manipulation) [page 1104]
- System Properties
- Additional User Parameters
- Procedure Parameters
- Function Parameters
- Table Parameter
- Array Parameters for Procedures and Functions
### 6.1.54 DATA_STATISTICS System View

Provides an overview of data statistics objects.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_STATISTICS_OID</td>
<td>BIGINT</td>
<td>Displays the data statistics object ID.</td>
</tr>
<tr>
<td>DATA_STATISTICS_TYPE</td>
<td>VARCHAR(9)</td>
<td>Displays the data statistics type.</td>
</tr>
<tr>
<td>DATA_STATISTICS_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the data statistics object schema.</td>
</tr>
<tr>
<td>DATA_STATISTICS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the data statistics name.</td>
</tr>
<tr>
<td>DATA_SOURCE_OBJECT_TYPE</td>
<td>VARCHAR(5)</td>
<td>Displays the data source object type, for example, TABLE.</td>
</tr>
<tr>
<td>DATA_SOURCE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the data source.</td>
</tr>
<tr>
<td>DATA_SOURCE_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of the data source.</td>
</tr>
<tr>
<td>DATA_SOURCE_COLUMN_NAMES</td>
<td>NVARCHAR(5000)</td>
<td>Displays the column names of the data source.</td>
</tr>
<tr>
<td>DATA_SOURCE_STORAGE_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the storage type of the data source: DEFAULT/EXTENDED.</td>
</tr>
<tr>
<td>DATA_SOURCE_PART_ID</td>
<td>BIGINT</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMMP</td>
<td>Displays the time when the data statistics object was created.</td>
</tr>
<tr>
<td>CONSTRAINT</td>
<td>VARCHAR(20)</td>
<td>Displays the constraint for the build algorithm for the data statistics object.</td>
</tr>
<tr>
<td>IS_PERSISTENT</td>
<td>VARCHAR(5)</td>
<td>Displays whether the content of the data statistics object is persistent or not: TRUE/FALSE.</td>
</tr>
<tr>
<td>REFRESH_TYPE</td>
<td>VARCHAR(7)</td>
<td>Displays the refresh type for the data statistics object.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>IS_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the data statistics object is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>LAST_REFRESH_TIME</td>
<td>TIMESTAMP</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>LAST_REFRESH_REASON</td>
<td>VARCHAR(19)</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>MINVALUE_NUMERIC</td>
<td>DOUBLE</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>MINVALUE_STRING</td>
<td>NVARCHAR(256)</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>MAXVALUE_NUMERIC</td>
<td>DOUBLE</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>MAXVALUE_STRING</td>
<td>NVARCHAR(256)</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>COUNT</td>
<td>BIGINT</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>DISTINCT_COUNT</td>
<td>BIGINT</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>NULL_COUNT</td>
<td>BIGINT</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>CREATE_MEMORY</td>
<td>BIGINT</td>
<td>Displays the memory size that the data statistics object uses in bytes. This value is set in the CREATE STATISTICS statement.</td>
</tr>
<tr>
<td>LAST_REFRESH_MEMORY</td>
<td>BIGINT</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CREATE_MEMORY_PERCENT</td>
<td>REAL</td>
<td>Displays the memory size, as a percentage of the data source size, to be used by the data statistics object specified in the CREATE STATISTICS statement</td>
</tr>
<tr>
<td>LAST_REFRESH_MEMORY_PERC</td>
<td>REAL</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>CREATE_BUCKETS</td>
<td>BIGINT</td>
<td>Displays the number of buckets specified in the CREATE STATISTICS statement</td>
</tr>
<tr>
<td>LAST_REFRESH_BUCKETS</td>
<td>BIGINT</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>CREATE_QERROR</td>
<td>REAL</td>
<td>Displays the parameter qerror for the data statistics object of type HISTOGRAM as specified in the CREATE STATISTICS statement</td>
</tr>
<tr>
<td>LAST_REFRESH_QERROR</td>
<td>REAL</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>CREATE_QTHETA</td>
<td>INTEGER</td>
<td>Displays the parameter qtheta for the data statistics object of type HISTOGRAM as specified at the CREATE STATISTICS statement</td>
</tr>
<tr>
<td>LAST_REFRESH_QTHETA</td>
<td>INTEGER</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>CREATE_ACCURACY</td>
<td>REAL</td>
<td>Displays the parameter accuracy for the data statistics object of type SKETCH as specified in the CREATE STATISTICS statement</td>
</tr>
<tr>
<td>LAST_REFRESH_ACCURACY</td>
<td>REAL</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>CREATE_PREFIX_BITS</td>
<td>INTEGER</td>
<td>Displays the parameter prefix bits for the data statistics object of type SKETCH as specified at the CREATE STATISTICS statement</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LAST_REFRESH_PREFIX_BITS</td>
<td>INTEGER</td>
<td>Deprecated. See the M_DATA_STATISTICS view for more information about this column.</td>
</tr>
<tr>
<td>CREATE_SAMPLE_SIZE</td>
<td>BIGINT</td>
<td>Displays the sample size, in bytes, to be used by the data statistics object of type SAMPLE specified in the CREATE STATISTICS statement.</td>
</tr>
<tr>
<td>CREATE_SAMPLE_SIZE_PERCENT</td>
<td>REAL</td>
<td>Displays the sample size, as a percentage of the data source size, to be used by the data statistics object of type SAMPLE specified in the CREATE STATISTICS statement.</td>
</tr>
<tr>
<td>VALID_FOR</td>
<td>VARCHAR(32)</td>
<td>Displays how the data statistics object may be used: ESTIMATION/DATA DEPENDENCY.</td>
</tr>
<tr>
<td>INVALIDATION_REASON</td>
<td>VARCHAR(32)</td>
<td>Displays the reason for invalidating a data statistics object:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NULL</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The object is valid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>USER ACTION</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The data statistics object is invalid because a user has disabled it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ALTER TABLE MOVE</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The data statistics object is disabled because a user relocated the table to a different storage space.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ALTER COLUMN DATA TYPE</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The data statistics object is disabled because the data type of the underlying column(s) no longer supports the statistics type.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE STATISTICS Statement (Data Definition) [page 815]
ALTER STATISTICS Statement (Data Definition) [page 484]
REFRESH STATISTICS Statement (Data Definition) [page 1082]
DROP STATISTICS Statement (Data Definition) [page 974]
M_DATA_STATISTICS System View [page 1851]
6.1.55 DATA_TYPES System View

Provides information about available SQL data types.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the data type ID.</td>
</tr>
<tr>
<td>TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the data type name.</td>
</tr>
<tr>
<td>COLUMN_SIZE</td>
<td>INTEGER</td>
<td>Displays the maximum size, in bytes, of the data type that this system can support.</td>
</tr>
<tr>
<td>LITERAL_PREFIX</td>
<td>CHAR(1)</td>
<td>Displays the ODBC 2.0 SQLGetTypeInfo VARCHAR. Character or characters used to prefix a literal, for example, a single quotation mark (‘) for character data types or 0x for binary data types; NULL is returned for data types where a literal prefix is not applicable.</td>
</tr>
<tr>
<td>LITERAL_SUFFIX</td>
<td>CHAR(1)</td>
<td>Displays the ODBC 2.0 SQLGetTypeInfo VARCHAR. Character or characters used to terminate a literal, for example, a single quotation mark (‘) for character data types; NULL is returned for data types where a literal suffix is not applicable.</td>
</tr>
<tr>
<td>CREATE_PARAMS</td>
<td>VARCHAR(16)</td>
<td>Displays the ODBC 2.0 SQLGetTypeInfo VARCHAR. A list of keywords, separated by commas, corresponding to each parameter that the application may specify in parentheses when using the name that is returned in the TYPE_NAME field.</td>
</tr>
<tr>
<td>NULLABLE</td>
<td>TINYINT</td>
<td>Displays whether the data type can accept NULL or not.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CASE_SENSITIVE</td>
<td>TINYINT</td>
<td>Displays the ODBC 2.0 SQLGetTypeInfo SMALLINT. Whether a character data type is case-sensitive in collations and comparisons.</td>
</tr>
<tr>
<td>SEARCHABLE</td>
<td>TINYINT</td>
<td>Displays the how the data type can be used in the WHERE clause.</td>
</tr>
<tr>
<td>UNSIGNED_ATTRIBUTE</td>
<td>TINYINT</td>
<td>Displays whether the attribute is signed or unsigned.</td>
</tr>
<tr>
<td>FIXED_PREC_SCALE</td>
<td>TINYINT</td>
<td>Displays the whether the data type has predefined fixed precision and scale (ODBC).</td>
</tr>
<tr>
<td>AUTO_UNIQUE_VALUE</td>
<td>TINYINT</td>
<td>Displays whether the data type is auto incrementing (ODBC).</td>
</tr>
<tr>
<td>LOCAL_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the ODBC 2.0 SQLGetTypeInfo VARCHAR. Localized version of the data source-dependent name of the data type.</td>
</tr>
<tr>
<td>MINIMUM_SCALE</td>
<td>SMALLINT</td>
<td>Displays the ODBC 2.0 SQLGetTypeInfo SMALLINT. The minimum scale of the data type on the data source.</td>
</tr>
<tr>
<td>MAXIMUM_SCALE</td>
<td>SMALLINT</td>
<td>Displays the ODBC 2.0 SQLGetTypeInfo SMALLINT. The maximum scale of the data type on the data source.</td>
</tr>
<tr>
<td>SQL_DATA_TYPE</td>
<td>SMALLINT</td>
<td>Displays the ODBC 3.0 SQLGetTypeInfo SMALLINT. The value of the SQL data type as it appears in the SQL_DESC_TYPE field of the descriptor. This column is the same as the DATATYPE column, except for interval and DATETIME data types.</td>
</tr>
<tr>
<td>SQL_DATETIME_SUB</td>
<td>SMALLINT</td>
<td>Displays the ODBC 3.0 SQLGetTypeInfo SMALLINT. When the value of SQL_DATATYPE is SQL_DATETIME or SQL_INTERVAL, this column contains the DATETIME/interval subcode.</td>
</tr>
</tbody>
</table>
### NUM_PREC_RADIX

**Data type**: INTEGER

Displays the ODBC 3.0 SQLGetTypeInfo INTEGER. In case of an approximate numeric data type, value is 2 to indicate that COLUMN SIZE specifies the number of bits. For exact numeric data types value 10 indicates that COLUMN SIZE specifies the number of decimal digits.

### INTERVAL_PRECISION

**Data type**: SMALLINT

Displays the ODBC 3.0 SQLGetTypeInfo SMALLINT. If the data type is an interval data type, then this column contains the value of the interval leading precision. Otherwise, this column is NULL.

### Related Information

- Data Types [page 33]
- Data Type Conversion Functions [page 416]
- Data Type Support
- Data Type Extension
- Scalar Data Types

### 6.1.56 DEPENDENCY_RULES System View

For internal use only. Provides a list of dependency rules in the system.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPENDENCY_RULE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of dependency rule.</td>
</tr>
<tr>
<td>DEPENDENCY_RULE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of dependency rule.</td>
</tr>
<tr>
<td>DEPENDENCY_RULE_TYPE</td>
<td>VARCHAR(6)</td>
<td>Displays the type of dependency rule.</td>
</tr>
</tbody>
</table>
### Related Information

DEPENDENCY_RULE_COLUMNS System View [page 1545]

### 6.1.57 DEPENDENCY_RULE_COLUMNS System View

For internal use only. Provides a list dependency rule columns in the system.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPENDENCY_RULE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the dependency rule.</td>
</tr>
<tr>
<td>DEPENDENCY_RULE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the dependency rule.</td>
</tr>
<tr>
<td>OBJECT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the table that contains the column.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table that contains the column.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
</tbody>
</table>

### Related Information

DEPENDENCY_RULES System View [page 1544]
6.1.58  DYNAMIC_RESULT_CACHE System View

Provides information about metadata objects that are enabled for a dynamic result cache.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the object ID.</td>
</tr>
<tr>
<td>CACHE_TYPE</td>
<td>VARCHAR(44)</td>
<td>Displays which type of cache is enabled for the object: FULL, PARTIAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(SELECTED COLUMNS), PARTIAL (FILTERED), PARTIAL (SELECTED COLUMNS, FILTERED).</td>
</tr>
<tr>
<td>CACHE_FILTER</td>
<td>NCLOB</td>
<td>Displays the filter condition specified to reduce the cache size.</td>
</tr>
<tr>
<td>CACHE_LOCATIONS</td>
<td>VARCHAR(5000)</td>
<td>Displays the cache entry locations.</td>
</tr>
</tbody>
</table>

Related Information

[**M_DYNAMIC_RESULT_CACHE System View**](page 1875)
[M_DYNAMIC_RESULT_CACHE_EXCLUSIONS System View](page 1877)
[DYNAMIC_RESULT_CACHE_INDEX_COLUMNS System View](page 1547)
[ALTER SYSTEM REMOVE RESULT CACHE ENTRY Statement (System Management)](page 561)
[ALTER SYSTEM CLEAR RESULT CACHE Statement (System Management)](page 522)
6.1.59 DYNAMIC_RESULT_CACHE_INDEX_COLUMNS System View

Provides information about the indexes of dynamic result caches.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the object ID.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the indexed column.</td>
</tr>
</tbody>
</table>

Related Information

DYNAMIC_RESULT_CACHE System View [page 1546]
M_DYNAMIC_RESULT_CACHE System View [page 1875]
M_DYNAMIC_RESULT_CACHE_EXCLUSIONS System View [page 1877]
ALTER SYSTEM REMOVE RESULT CACHE ENTRY Statement (System Management) [page 561]
ALTER SYSTEM CLEAR RESULT CACHE Statement (System Management) [page 522]

6.1.60 EFFECTIVE_APPLICATION_PRIVILEGES System View

Provides information about the application privileges of the specified user.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user for whom effective privileges are shown.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GRANTEE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema of the grantee</td>
</tr>
<tr>
<td>GRANTEE</td>
<td>NVARCHAR(256)</td>
<td>Displays the user or role that has the privilege.</td>
</tr>
<tr>
<td>GRANTEE_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the type of grant: USER/ROLE.</td>
</tr>
<tr>
<td>GRANTOR_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the role that provided the privilege.</td>
</tr>
<tr>
<td>GRANTOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the user or role that provided the privilege.</td>
</tr>
<tr>
<td>GRANTOR_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the grantor type: USER/ROLE.</td>
</tr>
<tr>
<td>PRIVILEGE</td>
<td>NVARCHAR(256)</td>
<td>Displays the privilege granted.</td>
</tr>
<tr>
<td>IS_GRANTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the privilege was granted with WITH GRANT OPTION or WITH ADMIN OPTION: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the privilege is valid or has become invalid due to implicit revoking: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

EFFECTIVE_PRIVILEGE_GRANTEES System View [page 1552]
EFFECTIVE_PRIVILEGES System View [page 1550]
EFFECTIVE_STRUCTURED_PRIVILEGES System View [page 1555]
PRIVILEGES System View [page 1611]
System Views for Verifying Users’ Authorization
### 6.1.61 EFFECTIVE_MASK_EXPRESSIONS System View

Provides information as to how data is exposed to certain users in terms of data masking.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name. An equal predicate is required.</td>
</tr>
<tr>
<td>ROOT_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object. An equal predicate is required.</td>
</tr>
<tr>
<td>ROOT_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name. An equal predicate is required.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name. An equal predicate is required.</td>
</tr>
<tr>
<td>EFFECTIVE_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the effective user whom the mask expression enforced on the underlying object it applies to.</td>
</tr>
<tr>
<td>DEPENDENT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the dependent column.</td>
</tr>
<tr>
<td>UNDERLYING_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the object at its current level.</td>
</tr>
<tr>
<td>UNDERLYING_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of the object at its current level.</td>
</tr>
<tr>
<td>UNDERLYING_OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type name of the object at its current level.</td>
</tr>
<tr>
<td>UNDERLYING_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column at its current level.</td>
</tr>
</tbody>
</table>
### Additional Information

This view requires an equal predicate on ROOT_SCHEMA_NAME, ROOT_OBJECT_NAME, ROOT_COLUMN_NAME, and USER_NAME.

### Related Information

- CREATE VIEW Statement (Data Definition) [page 904]
- ALTER VIEW Statement (Data Definition) [page 667]
- System Views for Verifying Users' Authorization

### 6.1.62 EFFECTIVE_PRIVILEGES System View

Provides the privileges of the specified user.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user for whom effective privileges are shown.</td>
</tr>
<tr>
<td>GRANTEE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema of the grantee</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GRANTEE</td>
<td>NVARCHAR(256)</td>
<td>Displays the user or role that has the privilege.</td>
</tr>
<tr>
<td>GRANTEE_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the grantee type: USER/ROLE.</td>
</tr>
<tr>
<td>GRANTOR_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the role that provided the privilege.</td>
</tr>
<tr>
<td>GRANTOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the user or role that provided the privilege.</td>
</tr>
<tr>
<td>GRANTOR_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the grantor type: USER/ROLE.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of the granted object, for example, TABLE, SCHEMA, and so on.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name the object belongs to.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of granted object.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>PRIVILEGE</td>
<td>NVARCHAR(256)</td>
<td>Displays the privilege granted.</td>
</tr>
<tr>
<td>IS_GRANTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the privilege was granted with WITH GRANT OPTION, WITH ADMIN OPTION: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the privilege is valid or has become invalid due to implicit revoking: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view requires an equal predicate on USER_NAME.

**Related Information**

- EFFECTIVE_APPLICATION_PRIVILEGES System View [page 1547]
- EFFECTIVE_PRIVILEGE_GRANTEES System View [page 1552]
- PRIVILEGES System View [page 1611]
6.1.63 EFFECTIVE_PRIVILEGE_GRANTEES System View

Provides information about who was granted (explicitly or implicitly via roles) a specified privilege.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANTEE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of role that has the privilege.</td>
</tr>
<tr>
<td>GRANTEE</td>
<td>NVARCHAR(256)</td>
<td>Displays the user or role that has the privilege.</td>
</tr>
<tr>
<td>GRANTEE_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the grantee type: USER/ROLE.</td>
</tr>
<tr>
<td>GRANTOR_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of role that provided the privilege.</td>
</tr>
<tr>
<td>GRANTOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the user or role that provided the privilege.</td>
</tr>
<tr>
<td>GRANTOR_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the grantor type: USER/ROLE.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>NVARCHAR(32)</td>
<td>Displays the type of the granted object, for example, TABLE, SCHEMA, and so on.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name the object belongs to.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of granted object.</td>
</tr>
<tr>
<td>PRIVILEGE</td>
<td>NVARCHAR(256)</td>
<td>Displays the privilege granted.</td>
</tr>
<tr>
<td>IS_GRANTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the privilege was granted with WITH GRANT OPTION or WITH ADMIN OPTION: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
Additional Information

This view requires an equal predicate on PRIVILEGE and OBJECT_TYPE.

Related Information

EFFECTIVE_PRIVILEGES System View [page 1550]
EFFECTIVE_APPLICATION_PRIVILEGES System View [page 1547]
PRIVILEGES System View [page 1611]
System Views for Verifying Users’ Authorization

6.1.64 EFFECTIVE_ROLES System View

Provides the roles of the current user.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user for whom effective roles are shown.</td>
</tr>
<tr>
<td>GRANTEE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema of the grantee.</td>
</tr>
<tr>
<td>GRANTEE</td>
<td>NVARCHAR(256)</td>
<td>Displays the user or role that has the role.</td>
</tr>
<tr>
<td>GRANTEE_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the grantee type: USER/ROLE.</td>
</tr>
<tr>
<td>GRANTOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the user who granted the role.</td>
</tr>
<tr>
<td>ROLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema of the role granted.</td>
</tr>
<tr>
<td>ROLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the role granted.</td>
</tr>
<tr>
<td>IS_GRANTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the role was granted with WITH ADMIN OPTION: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
Additional Information

This view requires an equal predicate on USER_NAME.

Related Information

EFFECTIVE_ROLE_GRANTEES System View [page 1554]  
ROLES System View [page 1646]  
CREATE ROLE Statement (Access Control) [page 802]  
ALTER ROLE Statement (Access Control) [page 475]  
DROP ROLE Statement (Access Control) [page 967]  
REVOKE Statement (Access Control) [page 1098]  
GRANTED_ROLES System View [page 1581]  
Database Roles  
Change a Role  
Delete a Role  
Database Role Details  
Assign Roles to Database Users  
System Views for Verifying Users' Authorization

6.1.65 EFFECTIVE_ROLE_GRANTEES System View

Provides information regarding the users and roles that the role is granted to.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the role for whom effective grantees are shown.</td>
</tr>
<tr>
<td>ROLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the role for whom effective grantees are shown.</td>
</tr>
<tr>
<td>GRANTEE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of grantee that has the role.</td>
</tr>
<tr>
<td>GRANTEE</td>
<td>NVARCHAR(256)</td>
<td>Displays the user or role that has the role.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
GRANTEE_TYPE | VARCHAR(4) | Displays the grantee type: USER/ROLE.
GRANTOR | NVARCHAR(256) | Displays the user that granted the role.
GRANTED_ROLE_SCHEMA_NAME | NVARCHAR(256) | Displays the schema name of granted role.
GRANTED_ROLE_NAME | NVARCHAR(256) | Displays the role granted.
IS_GRANTABLE | VARCHAR(5) | Displays whether the role was granted with WITH ADMIN OPTION: TRUE/FALSE.

**Related Information**

- EFFECTIVE_ROLES System View [page 1553]
- GRANTED_ROLES System View [page 1581]
- ROLES System View [page 1646]
- CREATE ROLE Statement (Access Control) [page 802]
- ALTER ROLE Statement (Access Control) [page 475]
- DROP ROLE Statement (Access Control) [page 967]
- REVOKE Statement (Access Control) [page 1098]
- Database Roles
- Change a Role
- Delete a Role
- Database Role Details
- Assign Roles to Database Users
- System Views for Verifying Users' Authorization

### 6.1.66 EFFECTIVESTRUCTURED_PRIVILEGES System View

Displays the structured privileges applied to an object.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the root schema name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ROOT_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>USER_ID</td>
<td>BIGINT</td>
<td>Displays the user ID.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object.</td>
</tr>
<tr>
<td>OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the object ID.</td>
</tr>
<tr>
<td>EFFECTIVE_FILTER</td>
<td>NCLOB</td>
<td>Displays the filter condition applied from all structured privileges.</td>
</tr>
<tr>
<td>STRUCTURED_PRIVILEGE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of a structured privilege created for an object.</td>
</tr>
<tr>
<td>STRUCTURED_PRIVILEGE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of a structured privilege created for an object.</td>
</tr>
<tr>
<td>STRUCTURED_PRIVILEGE_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the structured privilege.</td>
</tr>
<tr>
<td>STRUCTURED_PRIVILEGE_FILTER</td>
<td>NCLOB</td>
<td>Displays the filter condition provided by the structured privilege.</td>
</tr>
<tr>
<td>STRUCTURED_PRIVILEGE_DEFAULT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the default schema name of the structured privilege. This schema name is used for unqualified tables and views in a complex filter.</td>
</tr>
<tr>
<td>STRUCTURED_PRIVILEGE_STATUS</td>
<td>NVARCHAR(64)</td>
<td>Displays the status of a particular structured privilege: APPLIED, NOT GRANTED, NO MATCHING ATTRIBUTE, or NO FILTER VALUES FOUND.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view requires an equal predicate on ROOT_SCHEMA_NAME, ROOT_OBJECT_NAME, and USER_NAME.

**Related Information**

STRUCTURED_PRIVILEGES System View [page 1662]
6.1.67 ELEMENT_TYPES System View

Provides information about available multivalued types.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema containing the object defining or using the collection type.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table, view, or procedure that defines or uses the collection type.</td>
</tr>
<tr>
<td>OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the identifier of the table, view, or procedure that defines or uses the collection type.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the object type: TABLE, VIEW, or FUNCTION.</td>
</tr>
<tr>
<td>ELEMENT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table or view column or procedure parameter.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the identifier of the element type.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the name of the element type.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the length for a character string type or the precision for a numeric type in bytes.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Displays the scale for a numeric type.</td>
</tr>
<tr>
<td>MIN_CARDINALITY</td>
<td>INTEGER</td>
<td>Displays the lower bound of the array.</td>
</tr>
<tr>
<td>MAX_CARDINALITY</td>
<td>INTEGER</td>
<td>Displays the upper bound of the array.</td>
</tr>
<tr>
<td>ALLOW_DUPLICATES</td>
<td>VARCHAR(5)</td>
<td>Indicates if duplicates are allowed within one column: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

- Multi-valued Data Types [page 47]
- CREATE TYPE Statement (Procedural) [page 892]
- DROP TYPE Statement (Procedural) [page 983]
- Data Type Support

**6.1.68 ENCRYPTION_ROOT_KEYS System View**

Provides information about root keys.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT_KEY_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the usage type of the root key: PERSISTENCE, DPAPI, LOG, BACKUP.</td>
</tr>
<tr>
<td>ROOT_KEY_VERSION</td>
<td>INTEGER</td>
<td>Deprecated: The version of the root key.</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the creation time of the root key (UTC).</td>
</tr>
<tr>
<td>IS_CONSISTENT</td>
<td>VARCHAR(5)</td>
<td>Displays whether the key is consistent between persistence and the SSFS file: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
### Related Information

- Generate New Root Keys
- Activate Root Keys
- Changing Encryption Root Keys
- Change Root Keys
- Root Key Backup
- Back Up Root Keys
- Import Backed-Up Root Keys or LSS Backup Before Database Recovery
- SAP HANA Backup Encryption
- Disable Encryption
- ENCRYPTION_ROOT_KEYS_EXTRACT_KEYS Function (Security) [page 187]
- ENCRYPTION_ROOT_KEYS_HAS_BACKUP_PASSWORD Function (Security) [page 190]
- ALTER SYSTEM SET ENCRYPTION_ROOT_KEYS_BACKUP_PASSWORD Statement (System Management) [page 568]
- ALTER SYSTEM VALIDATE ENCRYPTION_ROOT_KEYS_BACKUP_PASSWORD Statement (System Management) [page 587]
- ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
- ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
- ALTER SYSTEM LOG ENCRYPTION Statement (System Management) [page 541]
- ALTER SYSTEM APPLICATION ENCRYPTION Statement (System Management) [page 512]
- M_ENCRYPTION_OVERVIEW System View [page 1881]
- M_PERSISTENCE_ENCRYPTION_STATUS System View [page 2043]
- Client-side Encryption Statements [page 1203]
- Encryption Configuration
- Managing Data Encryption in SAP HANA
- SQLScript Encryption
- Encryption Key Management
6.1.69  EPM_MODELS System View

Provides information about available EPM Models.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the EPM Model.</td>
</tr>
<tr>
<td>MODEL_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the EPM Model.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(256)</td>
<td>Displays the description of the EPM Model.</td>
</tr>
<tr>
<td>CDATA</td>
<td>NCLOB</td>
<td>Displays the definition of the EPM Model with JSON representation.</td>
</tr>
</tbody>
</table>

Related Information

EPM_QUERY_SOURCES System View [page 1560]
M_EPM_SESSIONS System View [page 1882]

6.1.70  EPM_QUERY_SOURCES System View

Provides information about available EPM query source.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the EPM query source.</td>
</tr>
<tr>
<td>QUERY_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the EPM query source.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(256)</td>
<td>Displays the description of the EPM query source.</td>
</tr>
<tr>
<td>CDATA</td>
<td>NCLOB</td>
<td>Displays the definition of the EPM query source with JSON representation.</td>
</tr>
</tbody>
</table>

**Related Information**

- [EPM_MODELS System View](#) [page 1560]
- [M_EPM_SESSIONS System View](#) [page 1882]

**6.1.71 EXPLAIN_CALL_PLANS System View**

Provides information about the compiled plan of a given procedure.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPERATOR_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the name of logical operator.</td>
</tr>
<tr>
<td>OPERATOR_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string or code (llvm, expression, ce) string, or se plan/operation in text format.</td>
</tr>
<tr>
<td>RETURN_TYPES</td>
<td>VARCHAR(5000)</td>
<td>Displays the list of return types.</td>
</tr>
<tr>
<td>INPUT_VALUES</td>
<td>NVARCHAR(5000)</td>
<td>Displays the list of values of input parameters.</td>
</tr>
<tr>
<td>OUTPUT_VALUES</td>
<td>NVARCHAR(5000)</td>
<td>Displays the list of values of output parameters.</td>
</tr>
<tr>
<td>EXECUTION_ENGINE</td>
<td>VARCHAR(32)</td>
<td>Displays the list of all involved execution frameworks.</td>
</tr>
<tr>
<td>DEFAULT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the default schema of each operator.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROCEDURE_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of database where procedure is defined.</td>
</tr>
<tr>
<td>PROCEDURE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of schema where procedure is defined.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of outermost procedure.</td>
</tr>
<tr>
<td>SQLSCRIPT_PLAN_ID</td>
<td>INTEGER</td>
<td>Displays the SE plan ID.</td>
</tr>
<tr>
<td>SQLSCRIPT_OPERATOR_ID</td>
<td>INTEGER</td>
<td>Displays the SE Operation ID.</td>
</tr>
<tr>
<td>SQLSCRIPT_OPERATOR_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the internal name used in SQLScript.</td>
</tr>
<tr>
<td>NESTED_STATEMENT_IDS</td>
<td>VARCHAR(5000)</td>
<td>Displays the concatenated statement position with respect to SQL inlining.</td>
</tr>
<tr>
<td>SQLSCRIPT_OPERATOR_COST</td>
<td>DOUBLE</td>
<td>Displays the estimated cost from the SQLScript optimizer.</td>
</tr>
<tr>
<td>OPERATOR_DETAILS</td>
<td>NCLOB</td>
<td>Displays additional information on the operator.</td>
</tr>
<tr>
<td>OPERATOR_ID</td>
<td>INTEGER</td>
<td>Displays the unique ID of an operator.</td>
</tr>
<tr>
<td>PARENT_OPERATOR_ID</td>
<td>INTEGER</td>
<td>Displays the operator ID of the parent of an operator.</td>
</tr>
<tr>
<td>LEVEL</td>
<td>INTEGER</td>
<td>Displays the level from the root operator.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the position in the parent operator.</td>
</tr>
<tr>
<td>STATEMENT_NAME</td>
<td>NVARCHAR(64)</td>
<td>Displays the internal EXPLAIN PLAN command.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the target node where the execution is expected to take place in a distributed system.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the target node where the execution is expected to take place in a distributed system.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the date and time when the EXPLAIN PLAN command was executed.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
</tbody>
</table>

**Additional Information**

The OPERATOR_NAME column can contain **column engine** and **row engine** operators, as shown in the following tables.

**Column engine operators:**

<table>
<thead>
<tr>
<th>Operator Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMN SEARCH</td>
<td>Displays the starting position of the column engine operators. OPERATORDETAILS lists the projected columns.</td>
</tr>
<tr>
<td>LIMIT</td>
<td>Displays the operator for limiting the number of output rows.</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>Displays the operator for sorting output rows.</td>
</tr>
<tr>
<td>HAVING</td>
<td>Displays the operator for filtering with predicates on top of grouping and aggregation.</td>
</tr>
<tr>
<td>GROUP BY</td>
<td>Displays the operator for grouping and aggregation.</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>Displays the operator for duplicate elimination.</td>
</tr>
<tr>
<td>FILTER</td>
<td>Displays the operator for filtering with predicates.</td>
</tr>
<tr>
<td>JOIN</td>
<td>Displays the operator for joining input relations.</td>
</tr>
<tr>
<td>COLUMN TABLE</td>
<td>Displays information about the accessed column table.</td>
</tr>
<tr>
<td>MULTIPROVIDER</td>
<td>Displays the operator for producing union-all of multiple results having the same grouping and aggregation.</td>
</tr>
</tbody>
</table>

**Row engine operators:**

<table>
<thead>
<tr>
<th>Operator Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW SEARCH</td>
<td>Displays the starting position of row engine operators. OPERATOR_DETAILS lists projected columns.</td>
</tr>
<tr>
<td>Operator Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LIMIT</td>
<td>Displays the operator for limiting number of output rows.</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>Displays the operator for sorting output rows.</td>
</tr>
<tr>
<td>HAVING</td>
<td>Displays the operator for filtering with predicates on top of grouping and aggregation.</td>
</tr>
<tr>
<td>GROUP BY</td>
<td>Displays the operator for grouping and aggregation.</td>
</tr>
<tr>
<td>MERGE AGGREGATION</td>
<td>Displays the operator for merging the results of multiple parallel grouping and aggregations.</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>Displays the operator for duplicate elimination.</td>
</tr>
<tr>
<td>FILTER</td>
<td>Displays the operator for filtering with predicates.</td>
</tr>
<tr>
<td>UNION ALL</td>
<td>Displays the operator for producing union-all of input relations.</td>
</tr>
<tr>
<td>MATERIALIZED UNION ALL</td>
<td>Displays the operator for producing union-all of input relations with intermediate result materialization.</td>
</tr>
<tr>
<td>BTREE INDEX JOIN</td>
<td>Displays the operator for joining input relations through B-tree index searches. A join type suffix can be added, for example, a B-tree index join for a left outer join is shown as BTREE INDEX JOIN (LEFT OUTER). A join without a join type suffix indicates an inner join.</td>
</tr>
<tr>
<td>CPBTREE INDEX JOIN</td>
<td>Displays the operator for joining input relations through CPB-tree index searches. A join type suffix can be added.</td>
</tr>
<tr>
<td>HASH JOIN</td>
<td>Displays the operator for joining input relations through probing a hash table built on the fly. A join type suffix can be added.</td>
</tr>
<tr>
<td>NESTED LOOP JOIN</td>
<td>Displays the operator for joining input relations through nested looping. A join type suffix can be added.</td>
</tr>
<tr>
<td>MIXED INVERTED INDEX JOIN</td>
<td>Displays the operator for joining an input relation of row store format with a column table without format conversion using an inverted index of the column table. A join type suffix can be added.</td>
</tr>
<tr>
<td>BTREE INDEX SEARCH</td>
<td>Displays the table access through a B-tree index search.</td>
</tr>
<tr>
<td>CPBTREE INDEX SEARCH</td>
<td>Displays the table access through a CPB-tree index search.</td>
</tr>
<tr>
<td>TABLE SCAN</td>
<td>Displays the table access through scanning.</td>
</tr>
<tr>
<td>AGGR TABLE</td>
<td>Displays the operator for aggregating a base table directly.</td>
</tr>
</tbody>
</table>
Related Information

EXPLAIN PLAN Statement (Data Manipulation) [page 992]
CALL Statement (Procedural) [page 715]
EXPLAIN PLAN for Call
CALL
Best Practices for Using SQLScript

6.1.72 EXPLAIN_PLAN_TABLE System View

Provides information about SQL query plan explanation results.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATEMENT_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the string specified as STATEMENT_NAME when executing the EXPLAIN PLAN command. This is used to distinguish plans from each other when there are multiple plans in EXPLAIN_PLAN_TABLE view.</td>
</tr>
<tr>
<td>OPERATOR_NAME</td>
<td>VARCHAR(5000)</td>
<td>Displays a column engine or row engine operator. See the Additional Information section within this topic for more information about the possible values in this column.</td>
</tr>
<tr>
<td>OPERATOR_DETAILS</td>
<td>NCLOB</td>
<td>Displays details of an operator including predicates and expressions used by the operator.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OPERATOR_PROPERTIES.</td>
<td>NCLOB</td>
<td>Displays the internal properties of an operator.</td>
</tr>
<tr>
<td>EXECUTION_ENGINE</td>
<td>VARCHAR(256)</td>
<td>Displays the execution engine where the plan operator is executed: COLUMN, ROW, or SQLScript.</td>
</tr>
<tr>
<td>DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the database name that the object belongs to.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name the dependent object belongs to.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the database tables and views accessed by an operator.</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>VARCHAR(5000)</td>
<td>Displays the table stored type: COLUMN or ROW.</td>
</tr>
<tr>
<td>TABLE_SIZE</td>
<td>DOUBLE</td>
<td>Displays the estimated input row count of an operator. This is available only for operators accessing tables and views directly.</td>
</tr>
<tr>
<td>OUTPUT_SIZE</td>
<td>DOUBLE</td>
<td>Estimated output row count of an operator.</td>
</tr>
<tr>
<td>SUBTREE_COST</td>
<td>DOUBLE</td>
<td>Displays the estimated cost based on estimated cardinality information. This value is used for the cost-based optimizer to choose the best plan. The value can also be a possible indicator to compare two different plans for the same query; the smaller the subtree cost, typically the better the performance.</td>
</tr>
<tr>
<td>OPERATOR_ID</td>
<td>INTEGER</td>
<td>Displays the unique ID of an operator. IDs are integers starting from 1.</td>
</tr>
<tr>
<td>PARENT_OPERATOR_ID</td>
<td>INTEGER</td>
<td>Displays the operator ID of the parent of an operator. The shape of a query plan is a tree and the topology of the tree can be reconstructed using OPERATOR_ID and PARENT_OPERATOR_ID. The PARENT_OPERATOR_ID of the root operator appears as NULL.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LEVEL</td>
<td>INTEGER</td>
<td>Displays the level from the root operator. The level of the root operator is 1, the level of a child of the root operator is 2, and so on. This can be utilized for output indentation.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the position in the parent operator. The position of the first child is 1, the position of the second child is 2, and so on.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(256)</td>
<td>Displays the host where the plan operator is generated.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port where the plan operator is generated.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the date and time when the EXPLAIN PLAN command was executed.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
</tbody>
</table>

### Additional Information

The OPERATOR_NAME column can contain **column engine** and **row engine** operators, as shown in the following tables.

**Column engine operators:**

<table>
<thead>
<tr>
<th>Operator Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLUMN SEARCH</td>
<td>Displays the starting position of column engine operators. OPERATOR_DETAILS lists the projected columns.</td>
</tr>
<tr>
<td>LIMIT</td>
<td>Displays the operator for limiting the number of output rows.</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>Displays the operator for sorting output rows.</td>
</tr>
<tr>
<td>HAVING</td>
<td>Displays the operator for filtering with predicates on top of grouping and aggregation.</td>
</tr>
<tr>
<td>GROUP BY</td>
<td>Displays the operator for grouping and aggregation.</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>Displays the operator for duplicate elimination.</td>
</tr>
<tr>
<td>Operator Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>FILTER</td>
<td>Displays the operator for filtering with predicates.</td>
</tr>
<tr>
<td>JOIN</td>
<td>Displays the operator for joining input relations.</td>
</tr>
<tr>
<td>COLUMN TABLE</td>
<td>Displays information about accessed column tables.</td>
</tr>
<tr>
<td>MULTIPROVIDER</td>
<td>Displays the operator for producing union-all of multiple results having the same grouping and aggregation.</td>
</tr>
</tbody>
</table>

Row engine operators:

<table>
<thead>
<tr>
<th>Operator Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW SEARCH</td>
<td>Displays the starting position of row engine operators. OPERATOR_DETAILS lists the projected columns.</td>
</tr>
<tr>
<td>LIMIT</td>
<td>Displays the operator for limiting the number of output rows.</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>Displays the operator for sorting output rows.</td>
</tr>
<tr>
<td>HAVING</td>
<td>Displays the operator for filtering with predicates on top of grouping and aggregation.</td>
</tr>
<tr>
<td>GROUP BY</td>
<td>Displays the operator for grouping and aggregation.</td>
</tr>
<tr>
<td>MERGE AGGREGATION</td>
<td>Displays the operator for merging the results of multiple parallel grouping and aggregations.</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>Displays the operator for duplicate elimination.</td>
</tr>
<tr>
<td>FILTER</td>
<td>Displays the operator for filtering with predicates.</td>
</tr>
<tr>
<td>UNION ALL</td>
<td>Displays the operator for producing union-all of input relations.</td>
</tr>
<tr>
<td>MATERIALIZED UNION ALL</td>
<td>Displays the operator for producing union-all of input relations with intermediate result materialization.</td>
</tr>
<tr>
<td>BTREE INDEX JOIN</td>
<td>Displays the operator for joining input relations through B-tree index searches. Join type suffix can be added, for example, B-tree index join for left outer join is shown as BTREE INDEX JOIN (LEFT OUTER). Join without join type suffix means inner join.</td>
</tr>
<tr>
<td>CPBTREE INDEX JOIN</td>
<td>Displays the operator for joining input relations through CPB-tree index searches. Join type suffix can be added.</td>
</tr>
<tr>
<td>Operator Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HASH JOIN</td>
<td>Displays the operator for joining input relations through probing hash table built on the fly. Join type suffix can be added.</td>
</tr>
<tr>
<td>NESTED LOOP JOIN</td>
<td>Displays the operator for joining input relations through nested looping. Join type suffix can be added.</td>
</tr>
<tr>
<td>MIXED INVERTED INDEX JOIN</td>
<td>Displays the operator for joining an input relation of row store format with a column table without format conversion using an inverted index of the column table. Join type suffix can be added.</td>
</tr>
<tr>
<td>BTREE INDEX SEARCH</td>
<td>Displays the table access through B-tree index search.</td>
</tr>
<tr>
<td>CPBTREE INDEX SEARCH</td>
<td>Displays the table access through CPB-tree index search.</td>
</tr>
<tr>
<td>TABLE SCAN</td>
<td>Displays the table access through scanning.</td>
</tr>
<tr>
<td>AGGR TABLE</td>
<td>Displays the operator for aggregating base table directly.</td>
</tr>
<tr>
<td>MONITOR SEARCH</td>
<td>Displays the monitoring view access through a search.</td>
</tr>
<tr>
<td>MONITOR SCAN</td>
<td>Displays the monitoring view access through scanning.</td>
</tr>
</tbody>
</table>

**Related Information**

- EXPLAIN PLAN Statement (Data Manipulation) [page 992]
- ALTER SYSTEM (PIN | UNPIN) SQL PLAN CACHE ENTRY Statement (System Management) [page 548]
- EXPLAIN PLAN for Call
- EXPLAIN PLAN for Table User-Defined Functions
### 6.1.73 FLEXIBLE_TABLES System View

Provides information about available flexible tables.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>IS_RECLAIMABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the garbage collection is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>DEFAULT_DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the name of the default data type.</td>
</tr>
<tr>
<td>DEFAULT_LENGTH</td>
<td>INTEGER</td>
<td>Displays the length of the default data type.</td>
</tr>
<tr>
<td>DEFAULT_SCALE</td>
<td>INTEGER</td>
<td>Displays the scale of the default data type.</td>
</tr>
<tr>
<td>HAS_AUTO_DATA_TYPE_PROMOTION</td>
<td>VARCHAR(5)</td>
<td>Displays whether the automatic type promotion is enabled: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

#### Related Information

- ALTER SYSTEM RECLAIM FLEXIBLE TABLES Statement (System Management) [page 552]
- TABLES System View [page 1669]
- CREATE TABLE Statement (Data Definition) [page 825]
- ALTER TABLE Statement (Data Definition) [page 589]
6.1.74 FULLTEXT_INDEXES System View

Provides information about fulltext indexes on table columns.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the fulltext index.</td>
</tr>
<tr>
<td>INDEX_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the fulltext index.</td>
</tr>
<tr>
<td>LANGUAGE_COLUMN</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name of the indexed table containing language indicators for the indexed documents.</td>
</tr>
<tr>
<td>MIME_TYPE_COLUMN</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name of the indexed table containing MIME type indicators for the indexed documents.</td>
</tr>
<tr>
<td>LANGUAGE_DETECTION</td>
<td>NVARCHAR(256)</td>
<td>Displays the set of languages to be considered for automatic language detection.</td>
</tr>
<tr>
<td>FAST_PREPROCESS</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not preprocessing is done with reduced functionality (for example, linguistic search is not available), which speeds up the indexing: TRUE/FALSE.</td>
</tr>
<tr>
<td>FUZZY_SEARCH_INDEX</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not a fuzzy search is performed with an additional index (faster search, but higher memory consumption): TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SEARCH_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the text-attributes do not store any DAFs (it is not possible to retrieve the HTML-converted or original data from the text-attribute): TRUE/FALSE. If the text-attribute is a shadow attribute (created via CREATE FULLTEXT INDEX), the source-attribute containing the original data is not affected by this flag.</td>
</tr>
<tr>
<td>IS_EXPLICIT</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the index is an explicit fulltext index (created by CREATE FULLTEXT INDEX): TRUE/FALSE.</td>
</tr>
<tr>
<td>FLUSH_AFTER_DOCUMENTS</td>
<td>INTEGER</td>
<td>Displays the used to store change-tracking behavior of the fulltext index.</td>
</tr>
<tr>
<td>FLUSH_EVERY_MINUTES</td>
<td>INTEGER</td>
<td>Displays the data used to store the change-tracking behavior of the fulltext index in minutes.</td>
</tr>
<tr>
<td>CONFIGURATION</td>
<td>NVARCHAR(256)</td>
<td>Displays that an explicit configuration file can be specified, for example, to enable named entity extraction.</td>
</tr>
<tr>
<td>INTERNAL_COLUMN_NAME</td>
<td>NVARCHAR(512)</td>
<td>Displays the name of the internal text-attribute. This value is only set if the fulltext index was not created using the TEXT data type.</td>
</tr>
<tr>
<td>PHRASE_INDEX_RATIO</td>
<td>REAL</td>
<td>Displays the float between 0.0 and 1.0 which indicates how much memory the phrase index may use as a percentage of the memory size of the fulltext index.</td>
</tr>
<tr>
<td>TEXT_MINING</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the text mining capabilities are enabled on the indexed column: TRUE/FALSE.</td>
</tr>
<tr>
<td>TEXT_ANALYSIS</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not an additional table is created that stores text analysis results: TRUE/FALSE. Normally, this parameter should be set to FALSE.</td>
</tr>
<tr>
<td>MIME_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the MIME type indicator for the indexed documents.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TOKENSeparated</td>
<td>NVARCHAR(256)</td>
<td>Displays the characters used as token separators.</td>
</tr>
<tr>
<td>TEXT_MINING_CONFIGURATION</td>
<td>NVARCHAR(256)</td>
<td>Displays the configuration that is used for text mining.</td>
</tr>
<tr>
<td>TEXT_MINING_CONFIGURATION_OVERLAY</td>
<td>NVARCHAR(5000)</td>
<td>Displays the data used to override certain parameters from the configuration specified in TEXT_MINING_CONFIGURATION without changing the configuration itself.</td>
</tr>
<tr>
<td>COMPRESSION_LEVEL</td>
<td>TINYINT</td>
<td>Displays whether additional compression is applied to the fulltext index. A value of 0 means no compression and a value of 1 or larger means that compression is applied. The larger the value, the higher the compression.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE FULLTEXT INDEX Statement (Data Definition) [page 748]
ALTER FULLTEXT INDEX Statement (Data Definition) [page 445]
DROP FULLTEXT INDEX Statement (Data Definition) [page 955]
M_FULLTEXT_COLUMN_STATISTICS System View [page 1916]
M_FULLTEXT_QUEUES System View [page 1917]
INDEXING_STATUS Function (Fulltext) [page 216]
INDEXING_ERROR_CODE Function (Fulltext) [page 214]
INDEXING_ERROR_MESSAGE Function (Fulltext) [page 215]
INDEXES System View [page 1585]
6.1.75 FULL_SYSTEM_INFO_DUMPS System View

Provides the FSIDs for the current database/tenant.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_NAME</td>
<td>NVARCHAR(512)</td>
<td>Displays the file name of the collection.</td>
</tr>
<tr>
<td>FILE_SIZE</td>
<td>BIGINT</td>
<td>Displays the file size in bytes.</td>
</tr>
<tr>
<td>FILE_MTIME</td>
<td>TIMESTAMP</td>
<td>Displays the date the file was modified.</td>
</tr>
</tbody>
</table>

6.1.76 FUNCTIONS System View

Provides information about available functions.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the function.</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the function.</td>
</tr>
<tr>
<td>FUNCTION_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the function.</td>
</tr>
<tr>
<td>SQL_SECURITY</td>
<td>VARCHAR(7)</td>
<td>Displays the SQL security setting of the function: DEFINER/INVOKER.</td>
</tr>
<tr>
<td>DEFAULT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the unqualified objects in the function.</td>
</tr>
<tr>
<td>INPUT_PARAMETER_COUNT</td>
<td>INTEGER</td>
<td>Displays the input type parameter count.</td>
</tr>
<tr>
<td>RETURN_VALUE_COUNT</td>
<td>INTEGER</td>
<td>Displays the return value type parameter count.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_UNICODE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the function contains Unicode: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_ENCRYPTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the function is encrypted: TRUE/FALSE.</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NCLOB</td>
<td>Displays the query string of the function.</td>
</tr>
<tr>
<td>FUNCTION_TYPE</td>
<td>VARCHAR(10)</td>
<td>Displays the type of function.</td>
</tr>
<tr>
<td>FUNCTION_USAGE_TYPE</td>
<td>VARCHAR(9)</td>
<td>Displays the usage type of the function: SCALAR, TABLE, AGGREGATE, or WINDOW.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the function is valid or not. This value becomes FALSE when its base objects are changed or dropped: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_HEADER_ONLY.</td>
<td>VARCHAR(5).</td>
<td>Displays whether the function is a header-only function: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_DETERMINISTIC</td>
<td>VARCHAR(5)</td>
<td>Displays whether the function is deterministic: TRUE/FALSE.</td>
</tr>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the owner of the function.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time of the function.</td>
</tr>
</tbody>
</table>

**Related Information**

- FUNCTION_PARAMETERS System View [page 1576]
- FUNCTION_PARAMETER_COLUMNS System View [page 1577]
- CREATE FUNCTION Statement (Procedural) [page 752]
- ALTER FUNCTION Statement (Procedural) [page 447]
- DROP FUNCTION Statement (Procedural) [page 956]
- Aggregate Functions [page 414]
- Array Functions [page 415]
- Data Type Conversion Functions [page 416]
- Datetime Functions [page 417]
- Fulltext Functions [page 418]
- Hierarchy Functions [page 418]
### 6.1.77 FUNCTION_PARAMETERS System View

Provides information about parameters for functions.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the function.</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the function.</td>
</tr>
<tr>
<td>FUNCTION_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the function.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the parameter name.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the data type ID.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the data type name.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the parameter length.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Displays the scale of the parameter.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the parameter.</td>
</tr>
</tbody>
</table>
# Related Information

FUNCTION_PARAMETER_COLUMNS System View [page 1577]
FUNCTIONS System View [page 1574]
Function Parameters
Array Parameters for Procedures and Functions

## 6.1.78 FUNCTION_PARAMETER_COLUMNS System View

Provides information about columns that are available for function table parameters.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the function.</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the function.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FUNCTION_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the function.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the parameter name.</td>
</tr>
<tr>
<td>PARAMETER_POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the parameter.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column in the table parameter.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the column in the table parameter.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the SQL data type name of the column.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Depending on the data type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Displays the number of characters for characters types</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Displays the maximum number of digits for numeric types</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Displays the number of characters for datetime types</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Displays the number of bytes for LOB types</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>For numeric data types, displays the maximum number of digits to the right of the decimal point. For time and timestamp data types, displays the decimal digits, which are defined as the number of digits to the right of the decimal point in the second’s component of the data.</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column can accept NULL values: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

FUNCTION_PARAMETERS System View [page 1576]
FUNCTIONS System View [page 1574]
Function Parameters
Array Parameters for Procedures and Functions
# 6.1.79 GEOCODE_INDEXES System View

Provides information about geocode indexes on tables.

## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the index name.</td>
</tr>
<tr>
<td>INDEX_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the index.</td>
</tr>
<tr>
<td>FLUSH_AFTER_ROWS</td>
<td>INTEGER</td>
<td>Used to store change-tracking behavior of the index.</td>
</tr>
<tr>
<td>FLUSH_EVERY_MINUTES</td>
<td>INTEGER</td>
<td>Used to store change-tracking behavior of the index.</td>
</tr>
</tbody>
</table>

## Related Information

- INDEXES System View [page 1585]
- M_INDEXING_QUEUES System View [page 1940]
- GEOCODE_INFO_CODES System View [Smart Data Quality]
- GEOCODE_STATISTICS System View [Smart Data Quality]
6.1.80 GRANTED_PRIVILEGES System View

Provides information about privileges and roles granted to users.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANTEE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema of the grantee.</td>
</tr>
<tr>
<td>GRANTEE</td>
<td>NVARCHAR(256)</td>
<td>Displays the user or role the privilege is granted to.</td>
</tr>
<tr>
<td>GRANTEE_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the grantee type: USER/ROLE.</td>
</tr>
<tr>
<td>GRANTOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the user who granted the privilege.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of the granted object like: TABLE, SCHEMA, and so on.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name the object belongs to.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of granted object.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>PRIVILEGE</td>
<td>NVARCHAR(256)</td>
<td>Displays the privilege granted.</td>
</tr>
<tr>
<td>IS_GRANTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the privilege was granted with WITH GRANT OPTION or WITH ADMIN OPTION: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the privilege is valid or has become invalid because of implicit revoking: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

- GRANT Statement (Access Control) [page 1010]
- CREATE STRUCTURED PRIVILEGE Statement (Access Control) [page 822]
- ALTER STRUCTURED PRIVILEGE Statement (Access Control) [page 491]
6.1.81 GRANTED_ROLES System View

Provides information about roles granted to users or other roles.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANTEE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema of the grantee.</td>
</tr>
<tr>
<td>GRANTEE</td>
<td>NVARCHAR(256)</td>
<td>Displays the user or role the role is granted to.</td>
</tr>
<tr>
<td>GRANTEE_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the grantee type: USER/ROLE.</td>
</tr>
<tr>
<td>ROLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema of the role granted.</td>
</tr>
<tr>
<td>ROLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the granted role.</td>
</tr>
<tr>
<td>GRANTOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the user who granted the role.</td>
</tr>
<tr>
<td>IS_GRANTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the role was granted with ADMIN OPTION: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_GRANTED_BY_LDAP</td>
<td>VARCHAR(5)</td>
<td>Displays whether the role is granted by LDAP: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
**Related Information**

- **GRANT Statement (Access Control) [page 1010]**
- **CREATE ROLE Statement (Access Control) [page 802]**
- **ALTER ROLE Statement (Access Control) [page 475]**
- **DROP ROLE Statement (Access Control) [page 967]**
- **REVOKE Statement (Access Control) [page 1098]**
- **ROLES System View [page 1646]**
- **EFFECTIVE_ROLE_GRANTEES System View [page 1554]**
- **EFFECTIVE_ROLES System View [page 1553]**
- **GRANTED_PRIVILEGES System View [page 1580]**
- **Prerequisites for Granting and Revoking Privileges and Roles**
- **System Views for Verifying Users' Authorization**
- **Database Roles**
- **Database Role Details**
- **Assign Roles to Database Users**
- **Change a Role**

### 6.1.82 GRAPH_WORKSPACES System View

Provides information about graph workspaces in the database.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the name of the schema the workspace resides in.</td>
</tr>
<tr>
<td>WORKSPACE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the workspace name.</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the workspace creation timestamp.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the workspace creator user name.</td>
</tr>
<tr>
<td>EDGE_SCHEMA_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the schema name of workspace edge table.</td>
</tr>
<tr>
<td>EDGE_TABLE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the workspace edge table name.</td>
</tr>
<tr>
<td>EDGE_SOURCE_COLUMN_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the workspace source column name in the edge table.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EDGE_TARGET_COLUMN_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the workspace target column name in the edge table.</td>
</tr>
<tr>
<td>EDGE_KEY_COLUMN_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the workspace key column name in the edge table.</td>
</tr>
<tr>
<td>VERTEX_SCHEMA_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the schema name of the workspace vertex table.</td>
</tr>
<tr>
<td>VERTEX_TABLE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the workspace vertex table name.</td>
</tr>
<tr>
<td>VERTEX_KEY_COLUMN_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the workspace key column name in the vertex table.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR (5)</td>
<td>Displays whether this workspace is valid: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE GRAPH WORKSPACE Statement (Data Definition) [page 760]
DROP GRAPH WORKSPACE Statement (Data Definition) [page 957]
Object Privileges (Reference)
6.1.83 HAS_NEEDED_SYSTEM_PRIV System View

For internal use only.

6.1.84 HAS_NEEDED_SYSTEM_PRIV_INCL_SYS_STAT System View

For internal use only.

6.1.85 HIERARCHY_OBJECTS System View

Provides the list of objects that hierarchy navigation functions can be run on.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name for the hierarchy.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the hierarchy.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type of the hierarchy: TABLE, VIEW, or FUNCTION.</td>
</tr>
<tr>
<td>OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the hierarchy.</td>
</tr>
</tbody>
</table>

**Related Information**

- Hierarchy Functions [page 418]
- SAP HANA Hierarchy Developer Guide
6.1.86  HINTS System View

Provides all available hints to be used in WITH HINT clauses.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HINT_NAME</td>
<td>VARCHAR(45)</td>
<td>Displays the hint name that can be used inside the WITH HINT() syntax.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(5000)</td>
<td>Displays a description of the hint.</td>
</tr>
</tbody>
</table>

Related Information

HINT Details [page 1134]
STATEMENT_HINTS System View [page 1661]
Using Hints to Query Data Snapshots
ALTER SYSTEM (ADD | ALTER | REMOVE) STATEMENT HINT Statement (System Management) [page 493]
ALTER SYSTEM (ENABLE | DISABLE) STATEMENT HINT Statement (System Management) [page 535]
Statement Hints
Managing and Monitoring SAP HANA Performance
NO_INLINE and INLINE Hints
ROUTE_TO Hint

6.1.87  INDEXES System View

Provides information about indexes on tables.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table that the index is on.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table that the index is on.</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the index.</td>
</tr>
<tr>
<td>INDEX_OID</td>
<td>BIGINT</td>
<td>Displays the object ID for the index.</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the index type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Row store index types: BTREE, BTREE_UNIQUE, CPBTREE, or CPBTREE_UNIQUE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Column store index types: INVERTED VALUE, INVERTED VALUE UNIQUE, INVERTED HASH, INVERTED INDIVIDUAL, FULLTEXT TEXT ANALYSIS, or GEOCODE.</td>
</tr>
<tr>
<td>CONSTRAINT</td>
<td>VARCHAR(32)</td>
<td>Displays the constraint on the index: UNIQUE, NOT_NULL_UNIQUE, or PRIMARY_KEY.</td>
</tr>
<tr>
<td>KEY_LENGTH</td>
<td>SMALLINT</td>
<td>Displays the length of the key.</td>
</tr>
<tr>
<td>BTREE_FILL_FACTOR</td>
<td>SMALLINT</td>
<td>Displays the B-tree fill factor from 50-100.</td>
</tr>
<tr>
<td>BTREE_SPLIT_TYPE</td>
<td>TINYINT</td>
<td>Displays the B-tree split type: SPLIT_PERCENT/SPLIT_INVOKER.</td>
</tr>
<tr>
<td>BTREE_SPLIT_POSITION</td>
<td>TINYINT</td>
<td>Displays the B-tree split position from 0-100.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the index was created.</td>
</tr>
<tr>
<td>STORAGE_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the partial index storage type. NULL is returned if it is not a partial index. For multistore partial index, the value can be a string of DEFAULT</td>
</tr>
<tr>
<td>LOAD_UNIT</td>
<td>VARCHAR(7)</td>
<td>Displays the load unit for the index. Valid values are: COLUMN, PAGE, DEFAULT, or NULL.</td>
</tr>
</tbody>
</table>
Related Information

INDEX_COLUMNS System View [page 1587]
M_INDEXING_QUEUES System View [page 1940]
CREATE INDEX Statement (Data Definition) [page 762]
ALTER INDEX Statement (Data Definition) [page 452]
RENAME INDEX Statement (Data Definition) [page 1091]
DROP INDEX Statement (Data Definition) [page 958]
Changing the Load Units for Indexes Using ALTER INDEX
Using Indexes in SAP HANA NSE
Index-Based Cell Access to Table Variables

6.1.88 INDEX_COLUMNS System View

Provides information about index columns.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the index name.</td>
</tr>
<tr>
<td>INDEX_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the index.</td>
</tr>
<tr>
<td>CONSTRAINT</td>
<td>VARCHAR(32)</td>
<td>Displays the type of constraint on the index: UNIQUE, NOT_NULL_UNIQUE,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PRIMARY_KEY; Geocode type of the column: GEOCODE COUNTRY, GEOCODE STATE,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GEOCODE COUNTY, GEOCODE CITY, GEOCODE DISTRICT, GEOCODE STREET, GEOCODE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDRESS, or GEOCODE.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the indexed column.</td>
</tr>
</tbody>
</table>
### INDEXES System View [page 1585]
- CREATE INDEX Statement (Data Definition) [page 762]
- ALTER INDEX Statement (Data Definition) [page 452]
- RENAME INDEX Statement (Data Definition) [page 1091]
- DROP INDEX Statement (Data Definition) [page 958]

### INVALID_CONNECT_ATTEMPTS System View

Provides the number of invalid connection attempts for a user between two successful connections.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user.</td>
</tr>
<tr>
<td>SUCCESSFUL_CONNECT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time the valid connection was attempted.</td>
</tr>
<tr>
<td>INVALID_CONNECT_ATTEMPTS</td>
<td>BIGINT</td>
<td>Displays the number of invalid connection attempts for this user since the last successful connection.</td>
</tr>
</tbody>
</table>

### Related Information

- CONNECT Statement (Session Management) [page 721]
- M_CONNECTIONS System View [page 1785]
- Ports and Connections
### 6.1.90 JWT_PROVIDERS System View

Lists all of the JWT providers configured in the SAP HANA database.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JWT_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the JWT provider.</td>
</tr>
<tr>
<td>ISSUER_NAME</td>
<td>NVARCHAR(512)</td>
<td>Displays the name in the iss claim of the JWT tokens provided by this provider.</td>
</tr>
<tr>
<td>EXTERNAL.IDENTITY_CLAIM</td>
<td>NVARCHAR(256)</td>
<td>Displays the claim provided in the JWT tokens to use when mapping the SAP HANA user to an external user name.</td>
</tr>
<tr>
<td>IS_CASESENSITIVE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the user mapping is checked case sensitive: TRUE/FALSE.</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>TINYINT</td>
<td>Displays the priority of the provider compared to other JWT providers with an identical issuer.</td>
</tr>
<tr>
<td>IS_USER_CREATION_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the provider is allowed to create a user implicitly: TRUE/FALSE.</td>
</tr>
<tr>
<td>USER_CREATION_USER_TYPE</td>
<td>VARCHAR(10)</td>
<td>Displays the type of user to create when user creation is enabled: STANDARD, RESTRICTED</td>
</tr>
<tr>
<td>USER_CREATION_USERGROUP</td>
<td>NVARCHAR(256)</td>
<td>Displays the user group a created user will be a member of.</td>
</tr>
</tbody>
</table>
### Related Information

CREATE JWT PROVIDER Statement (Access Control) [page 767]  
DROP JWT PROVIDER Statement (Access Control) [page 959]  
ALTER JWT PROVIDER Statement (Access Control) [page 455]  
JWT_USER_MAPPINGS System View [page 1591]  
Add a JWT Identity Provider

### 6.1.91 JWT_PROVIDER_CLAIMS System View

Lists the additional claims with its values for the JWT provider.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_CREATION_AUTHORIZA-</td>
<td>NVARCHAR(5)</td>
<td>Displays the authorization mode of the automatic user creation. Possible val-</td>
</tr>
<tr>
<td>TION_MODE</td>
<td></td>
<td>ues are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• NULL (automatic user creation is off)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LDAP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JWT_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the JWT provider.</td>
</tr>
<tr>
<td>CLAIM</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the claim.</td>
</tr>
<tr>
<td>OPERATION</td>
<td>NVARCHAR(64)</td>
<td>Specifies how the claim is used. Operation include: EQUALS, HAS MEMBER,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET APPLICATION USER.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(512)</td>
<td>Display the expected value of the claim in a JWT token.</td>
</tr>
</tbody>
</table>
Permissions

Users must have rights to the JWT provider to see the claims in this view.

Related Information

CREATE JWT PROVIDER Statement (Access Control) [page 767]
DROP JWT PROVIDER Statement (Access Control) [page 959]
ALTER JWT PROVIDER Statement (Access Control) [page 455]
JWT_USER_MAPPINGS System View [page 1591]

6.1.92 JWT_USER_MAPPINGS System View

Lists all of the user-JWT mappings configured in the SAP HANA database.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the SAP HANA user name.</td>
</tr>
<tr>
<td>JWT_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the JWT provider.</td>
</tr>
<tr>
<td>EXTERNAL_IDENTITY</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user known to the JWT provider.</td>
</tr>
</tbody>
</table>

Related Information

CREATE JWT PROVIDER Statement (Access Control) [page 767]
DROP JWT PROVIDER Statement (Access Control) [page 959]
ALTER JWT PROVIDER Statement (Access Control) [page 455]
JWT_PROVIDERS System View [page 1589]
6.1.93 KEY_MANAGEMENT_CONFIGURATIONS System View

Displays information about the existing key management configurations.

Structure

<table>
<thead>
<tr>
<th>Column</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the database name.</td>
</tr>
<tr>
<td>CONFIGURATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the key management configuration.</td>
</tr>
<tr>
<td>IS_ACTIVE</td>
<td>NVARCHAR(5)</td>
<td>Indicates whether this configuration is used by LSS to define access to SAP HANA's sensitive information.</td>
</tr>
<tr>
<td>TYPE</td>
<td>NVARCHAR(20)</td>
<td>Displays the type of external key management.</td>
</tr>
<tr>
<td>CLIENT_VERSION</td>
<td>NVARCHAR(12)</td>
<td>Displays the version of the driver software.</td>
</tr>
<tr>
<td>PROPERTIES</td>
<td>NCLOB</td>
<td>Displays a reduced version of the properties section of the &lt;settings&gt; JSON (the &quot;credentials&quot; object is omitted).</td>
</tr>
<tr>
<td>CONFIGURATION_ID</td>
<td>NVARCHAR(64)</td>
<td>Displays the ID that uniquely identifies a key management configuration.</td>
</tr>
</tbody>
</table>

6.1.94 LCM_PRODUCTS System View

Provides information about all installed product versions.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VENDOR_NAME</td>
<td>VARCHAR(30)</td>
<td>Displays the vendor name, for example, sap.com.</td>
</tr>
<tr>
<td>PRODUCT_NAME</td>
<td>VARCHAR(30)</td>
<td>Displays the technical name of the product.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VERSION</td>
<td>VARCHAR(30)</td>
<td>Displays the version number of the PPMS product version, for example, 1.0.</td>
</tr>
<tr>
<td>PPMS_ID</td>
<td>VARCHAR(30)</td>
<td>Displays the technical ID of the product version, for example, the version one identifying it in PPMS.</td>
</tr>
<tr>
<td>PRODUCT_DESCRIPTION</td>
<td>VARCHAR(30)</td>
<td>Displays the human-readable name of the product.</td>
</tr>
<tr>
<td>SP_STACK_ID</td>
<td>VARCHAR(30)</td>
<td>Displays the SP stack ID of the PPMS product version.</td>
</tr>
<tr>
<td>SP_STACK_DESCRIPTION</td>
<td>VARCHAR(30)</td>
<td>Displays the human-readable name or description of the SP stack.</td>
</tr>
</tbody>
</table>

Related Information

- LCM_PRODUCT_INSTANCES System View [page 1593]
- LCM_PRODUCT_INSTANCES_INCLUDED System View [page 1594]
- LCM_SOFTWARE_COMPONENTS System View [page 1595]
- LCM_SWID System View [page 1596]
- Using the SAP HANA Platform LCM Tools

6.1.95 LCM_PRODUCT_INSTANCES System View

Provides information about all installed product instances.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VENDOR_NAME</td>
<td>VARCHAR(30)</td>
<td>Displays the name of the vendor. For example, sap.com.</td>
</tr>
<tr>
<td>PRODUCT_NAME</td>
<td>VARCHAR(30)</td>
<td>Displays the technical name of the product.</td>
</tr>
</tbody>
</table>
### Related Information

**LCM_PRODUCTS System View** [page 1592]
**LCM_PRODUCT_INSTANCES_INCLUDED System View** [page 1594]
**LCM_SOFTWARE_COMPONENTS System View** [page 1595]
**LCM_SWID System View** [page 1596]
**Using the SAP HANA Platform LCM Tools**

### 6.1.96 LCM_PRODUCT_INSTANCES_INCLUDED System View

Provides information about all installed included product instances.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VENDOR_NAME</td>
<td>VARCHAR(30)</td>
<td>Displays the name of the vendor. For example, sap.com.</td>
</tr>
<tr>
<td>PRODUCT_NAME</td>
<td>VARCHAR(30)</td>
<td>Displays the technical name of the product.</td>
</tr>
<tr>
<td>INSTANCE_ID</td>
<td>VARCHAR(3)</td>
<td>Displays the instance key.</td>
</tr>
<tr>
<td>PPMS_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the PPMS 2.0 PI number.</td>
</tr>
<tr>
<td>INSTANCE_DESCRIPTION</td>
<td>VARCHAR(64)</td>
<td>Displays the human-readable name of the product instance.</td>
</tr>
<tr>
<td>MASTER_PRODUCT_NAME</td>
<td>VARCHAR(30)</td>
<td>Displays the technical name of the including product.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>MASTER_INSTANCE_ID</td>
<td>VARCHAR(3)</td>
<td>Displays the instance key of the including product instance.</td>
</tr>
<tr>
<td>MASTER_PPMS_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the PPMS 2.0 PI number.</td>
</tr>
</tbody>
</table>

Related Information

LCM_PRODUCTS System View [page 1592]
LCM_PRODUCT_INSTANCES System View [page 1593]
LCM_SOFTWARE_COMPONENTS System View [page 1595]
LCM_SWID System View [page 1596]
Using the SAP HANA Platform LCM Tools

6.1.97 LCM_SOFTWARE_COMPONENTS System View

Provides all installed software component versions.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VENDOR_NAME</td>
<td>VARCHAR(30)</td>
<td>Displays the name of the vendor, for example, sap.com.</td>
</tr>
<tr>
<td>COMPONENT_NAME</td>
<td>VARCHAR(30)</td>
<td>Displays the technical name of the software component version.</td>
</tr>
<tr>
<td>VERSION</td>
<td>VARCHAR(30)</td>
<td>Displays the version of the delivery unit (=SWC), for example, 1.</td>
</tr>
<tr>
<td>VERSION_SP</td>
<td>VARCHAR(16)</td>
<td>Displays the number of the SP, for example, 3.</td>
</tr>
<tr>
<td>VERSION_PATCH</td>
<td>VARCHAR(16)</td>
<td>Displays the number of the patch, for example, 37.</td>
</tr>
</tbody>
</table>
### Related Information

- LCM_PRODUCT_INSTANCES_INCLUDED System View [page 1594]
- LCM_PRODUCTS System View [page 1592]
- LCM_PRODUCT_INSTANCES System View [page 1593]
- LCM_SWID System View [page 1596]
- Using the SAP HANA Platform LCM Tools

#### 6.1.98 LCM_SWID System View

Provides all software ID tags for installed software.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWID</td>
<td>VARCHAR(36)</td>
<td>Displays the SWID tag ID.</td>
</tr>
<tr>
<td>SWID_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the SWID name.</td>
</tr>
<tr>
<td>ENTITY_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the entity name (Vendor).</td>
</tr>
</tbody>
</table>
### Related Information

* LCM_SOFTWARE_COMPONENTS System View [page 1595]
* LCM_PRODUCT_INSTANCES_INCLUDED System View [page 1594]
* LCM_PRODUCTS System View [page 1592]
* LCM_PRODUCT_INSTANCES System View [page 1593]
* Using the SAP HANA Platform LCM Tools

### 6.1.99 LDAP_PROVIDERS System View

Lists all LDAP providers.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the LDAP provider.</td>
</tr>
<tr>
<td>LDAP_PROVIDER_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the LDAP provider.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time.</td>
</tr>
<tr>
<td>IS_DEFAULT</td>
<td>VARCHAR(5)</td>
<td>Displays whether the the LDAP provider is the default provider: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_SSL_USED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the connection to the configured LDAP Server uses SSL: TRUE/FALSE. If SSL ON is specified for the provider, or if USER LOOKUP URL or NESTED GROUP LOOKUP URL use 'ldaps://', the value is TRUE. Otherwise, the value is FALSE.</td>
</tr>
<tr>
<td>IS_PROVIDER_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the LDAP provider is enabled for use: TRUE/FALSE.</td>
</tr>
<tr>
<td>ATTRIBUTE_DN</td>
<td>VARCHAR(256)</td>
<td>Displays the LDAP attribute to provide the LDAP distinguished name of a user.</td>
</tr>
<tr>
<td>ATTRIBUTE_MEMBER_OF</td>
<td>NVARCHAR(256)</td>
<td>Displays the LDAP attribute that provides the list of LDAP groups that the user is a member of.</td>
</tr>
<tr>
<td>USER_TYPE_FOR_LDAP_USER_CREA-</td>
<td>VARCHAR(10)</td>
<td>Displays the type of user to create when LDAP user creation is enabled. NULL if LDAP user creation is not enabled: STANDARD, RESTRICTED, NULL</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE LDAP PROVIDER Statement (Access Control) [page 772]
ALTER LDAP PROVIDER Statement (Access Control) [page 458]
DROP LDAP PROVIDER Statement (Access Control) [page 962]
VALIDATE LDAP PROVIDER Statement (Access Control) [page 1195]
LDAP_PROVIDER_URLS System View [page 1599]
LDAP_USERS System View [page 1599]
LDAP Provider Configuration (Reference)
LDAP_PROVIDER_URLS System View [page 1599]
LDAP_USERS System View [page 1599]
6.1.100  LDAP_PROVIDER_URLS System View

Lists all LDAP provider URLs.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the LDAP provider.</td>
</tr>
<tr>
<td>URL TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the LDAP provider URL type.</td>
</tr>
<tr>
<td>URL</td>
<td>NVARCHAR(4096)</td>
<td>Displays the URL for the LDAP provider.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE LDAP PROVIDER Statement (Access Control) [page 772]
ALTER LDAP PROVIDER Statement (Access Control) [page 458]
DROP LDAP PROVIDER Statement (Access Control) [page 962]
VALIDATE LDAP PROVIDER Statement (Access Control) [page 1195]
LDAP_PROVIDERS System View [page 1597]
LDAP_USERS System View [page 1599]
LDAP Provider Configuration (Reference)

6.1.101  LDAP_USERS System View

Shows information about the users using LDAP authorization.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
DN | NVARCHAR(1024) | Displays the distinguished name of the user in the LDAP server.
LAST_AUTHORIZATION_REFRESH_TIME | TIMESTAMP | Displays the time of the last successful LDAP authorization refresh.

**Related Information**

CREATE LDAP PROVIDER Statement (Access Control) [page 772]
ALTER LDAP PROVIDER Statement (Access Control) [page 458]
DROP LDAP PROVIDER Statement (Access Control) [page 962]
VALIDATE LDAP PROVIDER Statement (Access Control) [page 1195]
LDAP_PROVIDER_URLS System View [page 1599]
LDAP_PROVIDERS System View [page 1597]
Configure LDAP Authentication and Authorization
LDAP Provider Configuration (Reference)

**6.1.102 LIBRARIES System View**

Provides information about available public language libraries.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the library.</td>
</tr>
<tr>
<td>LIBRARY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the library.</td>
</tr>
<tr>
<td>LIBRARY_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the library.</td>
</tr>
<tr>
<td>DEFAULT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the unqualified objects in the library.</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NCLOB</td>
<td>Displays the query string for the library.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LIBRARY_TYPE</td>
<td>VARCHAR(20)</td>
<td>Displays the language of the library: SQLSCRIPT, BUILTIN.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the library is valid: TRUE when base objects are valid and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the definition is valid. FALSE when the base objects for the library are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>changed or dropped.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time for the library.</td>
</tr>
</tbody>
</table>

**Related Information**

LIBRARY_MEMBERS System View [page 1601]  
CREATE LIBRARY Statement (SQLScript) [page 770]  
ALTER LIBRARY Statement (SQLScript) [page 462]  
DROP LIBRARY Statement (SQLScript) [page 961]  
Library Members  
User-Defined Libraries  
Built-In Libraries  
Library Member Functions and Variables

### 6.1.103 LIBRARY_MEMBERS System View

Provides member information for SQLScript user-defined libraries.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the library.</td>
</tr>
<tr>
<td>LIBRARY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the library.</td>
</tr>
<tr>
<td>LIBRARY_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the library.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MEMBER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the library member name.</td>
</tr>
<tr>
<td>MEMBER_TYPE</td>
<td>VARCHAR(9)</td>
<td>Displays the member type: VARIABLE, PROCEDURE, or FUNCTION.</td>
</tr>
<tr>
<td>ACCESS_MODE</td>
<td>VARCHAR(9)</td>
<td>Displays the access mode of the library member:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PUBLIC when the member is accessible without restriction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PRIVATE when the member is accessible within the library.</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NCLOB</td>
<td>Displays the definition of the library member.</td>
</tr>
<tr>
<td>PRAGMAS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the SQLScript user-defined test library pragmas.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the comments defined on user-defined libraries and members.</td>
</tr>
</tbody>
</table>

**Related Information**

- LIBRARIES System View [page 1600]
- SAP HANA SQLScript Reference
- CREATE LIBRARY Statement (SQLScript) [page 770]
- ALTER LIBRARY Statement (SQLScript) [page 462]
- DROP LIBRARY Statement (SQLScript) [page 961]
- User-Defined Libraries
- Library Members
- Built-In Libraries
- Library Member Functions and Variables
6.1.104 OBJECTS System View

Provides information about available objects.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_CATEGORY</td>
<td>VARCHAR(1)</td>
<td>Displays the object category of the object. The default is NULL.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the object.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type.</td>
</tr>
<tr>
<td>OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the object ID.</td>
</tr>
</tbody>
</table>

Related Information

OBJECT_DEPENDENCIES System View [page 1604]
OBJECT_PRIVILEGES System View [page 1605]
M_OBJECT_LOCKS System View [page 2032]
M_OBJECT_LOCK_STATISTICS System View [page 2033]
M_OBJECT_LOCK_STATISTICS_RESET System View [page 2035]
Managing Memory by Object Usage
Classification of Authorization Dependencies Between Objects
Import Schemas, Tables, and Other Catalog Objects
Export Schemas, Tables, and Other Catalog Objects
Import and Export of Encrypted SQLScript Objects
M_MEMORY_OBJECTS System View [page 2008]
6.1.105 OBJECT_DEPENDENCIES System View

Provides information about dependencies between objects, such as which views refer to a specific table.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the database name of the base objects.</td>
</tr>
<tr>
<td>BASE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the base object.</td>
</tr>
<tr>
<td>BASE_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of the base object.</td>
</tr>
<tr>
<td>BASE_OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of the base object.</td>
</tr>
<tr>
<td>DEPENDENT_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the database name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of the base dependency.</td>
</tr>
<tr>
<td>DEPENDENCY_TYPE</td>
<td>INTEGER</td>
<td>Displays the type of dependency between the base and dependent objects:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. Direct dependency (for example, the view uses a table)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Indirect dependency that is referenced via direct dependency (for</td>
</tr>
<tr>
<td></td>
<td></td>
<td>example, the view references a table via another view)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. SAP internal (dependency to an implicit created object in SQLScript)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. SAP internal (dependency to an implicit created object in SQLScript)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. Dependency via a referential constraint</td>
</tr>
</tbody>
</table>

SAP HANA SQL Reference Guide for SAP HANA Platform

System Views Reference
Related Information

OBJECTS System View [page 1603]
OBJECT_PRIVILEGES System View [page 1605]
M_TEMPORARY_OBJECT_DEPENDENCIES System View [page 2234]
Object Dependencies View Examples
SYS.OBJECT_DEPENDENCIES
Reduce Dependencies Between Statements
Classification of Authorization Dependencies Between Objects
Object Privileges
Import Tables and Other Catalog Objects
Export Tables and Other Catalog Objects
Toolbar Options in the Authorization Dependency Viewer

6.1.106 OBJECT_PRIVILEGES System View

Provides information about the types of objects and privileges that can be granted to those types of objects.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type.</td>
</tr>
<tr>
<td>PRIVILEGE</td>
<td>VARCHAR(40)</td>
<td>Displays the privilege that can be granted to this type of object.</td>
</tr>
</tbody>
</table>

Related Information

Object Privileges
Object Privileges (Reference)
Privileges
Managing Privileges
OBJECTS System View [page 1603]
OBJECT_DEPENDENCIES System View [page 1604]
PRIVILEGES System View [page 1611]
6.1.107 OWNERSHIP System View

Provides owner information for the objects available to the user who is querying the view.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the object.</td>
</tr>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object owner.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type.</td>
</tr>
<tr>
<td>OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the object ID.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application used for object creation.</td>
</tr>
</tbody>
</table>

Related Information

OBJECTS System View [page 1603]
Object Privileges

6.1.108 PARTITIONED_TABLES System View

Provides general partitioning information for all partitions of a table.

Structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LEVEL_1_EXPRESSION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the partitioning expression for first-level partitions.</td>
</tr>
<tr>
<td>LEVEL_1_COUNT</td>
<td>INTEGER</td>
<td>Displays the total number of partitions at the first level.</td>
</tr>
<tr>
<td>LEVEL_1_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the partitioning type at the first level: HASH, RANGE, RANGE_BIN, REPLICATE, or RANGE HETEROGENEOUS.</td>
</tr>
<tr>
<td>LEVEL_2_EXPRESSION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the partitioning expression for second-level partitions.</td>
</tr>
<tr>
<td>LEVEL_2_COUNT</td>
<td>INTEGER</td>
<td>Displays the total number of partitions at the second level.</td>
</tr>
<tr>
<td>LEVEL_2_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the partitioning type at the second level: HASH, RANGE, RANGE_HETEROGENEOUS, or HASH_HETEROGENEOUS. Heterogeneous indicates an unbalanced partitioning scheme.</td>
</tr>
<tr>
<td>CROSS_STORAGE_UNIQUE_CONSTRAINTS</td>
<td>VARCHAR(5)</td>
<td>Displays whether constraint checking is done across stores: TRUE/FALSE.</td>
</tr>
<tr>
<td>PARTITIONING_ON_NON_PRIMARY_KEY_COLUMNS</td>
<td>VARCHAR(5)</td>
<td>Displays whether partitioning is allowed on a non-primary key column: TRUE/FALSE. This value is only relevant for tables with time-selection partitioning.</td>
</tr>
<tr>
<td>PRIMARY_KEY_UPDATE</td>
<td>VARCHAR(5)</td>
<td>Displays whether UPDATE statements are allowed on primary key columns: TRUE/FALSE.</td>
</tr>
<tr>
<td>DYNAMIC_RANGE_THRESHOLD</td>
<td>BIGINT</td>
<td>Displays the threshold after which a new partition is created dynamically.</td>
</tr>
<tr>
<td>EXTENDED_STORAGE_ENABLE_DELTA</td>
<td>VARCHAR(5)</td>
<td>Displays whether extended storage is enabled to use a delta store: TRUE, FALSE, or NULL. NULL means that the system couldn’t retrieve the status from extended storage.</td>
</tr>
</tbody>
</table>
Additional Information

If you are using the dynamic tiering feature, then this monitoring view shows information for the extended storage corresponding to the connected SAP HANA database. The partitioning type is at the first level. Valid values are: HASH, RANGE, ROUNDROBIN, or REPLICATE.

You need the DATA ADMIN system privilege see information for tables owned by other users, unless you have SELECT permission on those tables.

Related Information

- TABLE_PARTITIONS System View [page 1681]
- M_TABLE_PARTITIONS System View [page 2209]
- M_TABLE_PARTITION_STATISTICS System View [page 2207]
- M_CS_PARTITIONS System View - Deprecated [page 1833]
- TABLES System View [page 1669]
- M_TABLES System View [page 2200]
- Table Partitioning
  - Partition a Non-Partitioned Table
  - Change a Partitioned Table into a Non-Partitioned Table
  - Move Table Partitions to Another Host
  - Optimize Table Partitioning
  - NSE-Enabled Partitioned Tables
- Heterogeneous Create Partition Clauses [page 869]

6.1.109 PERSISTENCE_HISTORY System View

Records the database version history.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGE_ID</td>
<td>INTEGER</td>
<td>Displays the order of database installations.</td>
</tr>
<tr>
<td>CHANGE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the database was upgraded or recovered to the current version.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VERSION</td>
<td>VARCHAR(32)</td>
<td>Displays the version.</td>
</tr>
<tr>
<td>SYSTEM_ID</td>
<td>VARCHAR(3)</td>
<td>Displays the system ID of the SAP HANA instance.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name of the catalog master. Only users with INIFILE ADMIN or CATALOG READ privileges are able to view this column.</td>
</tr>
</tbody>
</table>

Related Information

SET HISTORY SESSION Statement (Session Management) [page 1165]
M_BACKUP_HISTORY_BROKEN System View [page 1756]
M_JOB_HISTORY_INFO System View [page 1947]
Persistent Memory
SAP HANA History Tables
View Change History
View Redistribution Execution History

6.1.110 PINNED_SQL_PLANS System View

Provides information on currently pinned SQL plans and corresponding hints.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PINNED_PLAN_ID</td>
<td>BIGINT</td>
<td>Displays the pinned ID.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user who prepared the plan.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SESSION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the session user name who prepared the plan. This is the same value as displayed in the USER_NAME column in the M_CONNECTIONS monitored view.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name that the SQL plan belongs to.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the MD5 hash value.</td>
</tr>
<tr>
<td>IS_INTERNAL</td>
<td>TINYINT</td>
<td>Displays whether the plan is executed from a database internal connection or a remote connection: TRUE/FALSE.</td>
</tr>
<tr>
<td>ABAP_VARCHAR_MODE</td>
<td>TINYINT</td>
<td>Displays whether the value ABAP VARCHAR mode is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>PIN_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the plan was pinned.</td>
</tr>
<tr>
<td>LAST_MODIFY_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the plan was modified.</td>
</tr>
<tr>
<td>HINT_STRING</td>
<td>NCLOB</td>
<td>Displays the text string to indicate a hint was applied.</td>
</tr>
<tr>
<td>SESSION_PROPERTIES</td>
<td>NVARCHAR(500)</td>
<td>Displays the composed variables from the session context.</td>
</tr>
</tbody>
</table>

**Related Information**

ALTER SYSTEM (PIN | UNPIN) SQL PLAN CACHE ENTRY Statement (System Management) [page 548]
6.1.111 PRIVILEGES System View

Provides information about available privileges.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the privilege name.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the privilege type: SYSTEM-PRIVILEGE, OBJECTPRIVILEGE, PACKAGEPRIVILEGE, or APPLICATIONPRIVILEGE</td>
</tr>
</tbody>
</table>

Related Information

- Privileges
- Application Privileges
- Object Privileges
- Package Privileges
- System Privileges
- Analytic Privileges
- Managing Privileges
- GRANTED_PRIVILEGES System View [page 1580]
- Assign Privileges to a User
- OBJECT_PRIVILEGES System View [page 1605]
- GRANT Statement (Access Control) [page 1010]
- SQLSCRIPT_LOGGING Privilege
6.1.112 PROCEDURES System View

Provides information about available stored procedures.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the stored procedure.</td>
</tr>
<tr>
<td>SQL_SECURITY</td>
<td>VARCHAR(7)</td>
<td>Displays the SQL security setting of the stored procedure: DEFINER/INVOKER</td>
</tr>
<tr>
<td>DEFAULT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the unqualified objects in the procedure.</td>
</tr>
<tr>
<td>INPUT_PARAMETER_COUNT</td>
<td>INTEGER</td>
<td>Displays the input type parameter count.</td>
</tr>
<tr>
<td>OUTPUT_PARAMETER_COUNT</td>
<td>INTEGER</td>
<td>Displays the output type parameter count.</td>
</tr>
<tr>
<td>INOUT_PARAMETER_COUNT</td>
<td>INTEGER</td>
<td>Displays the in/out type parameter count.</td>
</tr>
<tr>
<td>RESULT_SET_COUNT</td>
<td>INTEGER</td>
<td>Displays the result set count.</td>
</tr>
<tr>
<td>IS_UNICODE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the stored procedure contains Unicode: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_ENCRYPTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the stored procedure is encrypted: TRUE/FALSE.</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NCLOB</td>
<td>Displays the query string of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_TYPE</td>
<td>VARCHAR(10)</td>
<td>Displays the type of the stored procedure.</td>
</tr>
<tr>
<td>READ_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the procedure is read-only: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the procedure is valid: TRUE/FALSE. A procedure become invalid when its base objects are changed or dropped.</td>
</tr>
<tr>
<td>IS_HEADER_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the procedure is a header only procedure: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_DETERMINISTIC</td>
<td>VARCHAR(5)</td>
<td>Displays whether the stored procedure is deterministic: TRUE/FALSE.</td>
</tr>
<tr>
<td>HAS_TRANSACTION_CONTROL_STATEMENTS</td>
<td>VARCHAR(5)</td>
<td>Displays whether the procedure has transaction control statements: TRUE/FALSE.</td>
</tr>
<tr>
<td>AUTO_COMMITDDL</td>
<td>VARCHAR(5)</td>
<td>Displays whether the stored procedure runs with autocommit DDL enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the owner of the stored procedure.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time of the procedure.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE PROCEDURE Statement (Procedural) [page 776]
ALTER PROCEDURE Statement (Procedural) [page 463]
DROP PROCEDURE Statement (Procedural) [page 963]
Procedural Statements [page 1204]
PROCEDURE_OBJECTS System View [page 1614]
PROCEDURE_PARAMETERS System View [page 1615]
PROCEDURE_PARAMETER_COLUMNS System View [page 1616]
PROCEDURE_ROUTES System View [page 1617]
Procedures
CREATE PROCEDURE
ALTER PROCEDURE
DROP PROCEDURE
Deterministic Procedures
Procedure Metadata
Procedure Calls
Procedure Result Cache
SYS.PROCEDURES
6.1.113 PROCEDURE_OBJECTS System View

Contains the results of the system procedure GET_PROCEDURE_OBJECTS.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(128)</td>
<td>Displays the schema name of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(128)</td>
<td>Displays the procedure name of the stored procedure.</td>
</tr>
<tr>
<td>OBJECT_SCHEMA</td>
<td>NVARCHAR(128)</td>
<td>Displays the schema name of the object in the stored procedure code.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(128)</td>
<td>Displays the object name of the object in the stored procedure code.</td>
</tr>
<tr>
<td>OBJECT_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the type ID of the object in the stored procedure code.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of the object in the stored procedure code.</td>
</tr>
<tr>
<td>START_POSITION</td>
<td>INTEGER</td>
<td>Displays the start position of the object in the stored procedure code.</td>
</tr>
<tr>
<td>END_POSITION</td>
<td>INTEGER</td>
<td>Displays the end position of the object in the stored procedure code.</td>
</tr>
</tbody>
</table>

Related Information

PROCEDURES System View [page 1612]
PROCEDURE_PARAMETERS System View [page 1615]
PROCEDURE_PARAMETER_COLUMNS System View [page 1616]
PROCEDURE_ROUTES System View [page 1617]
SYS.PROCEDURES
6.1.114 PROCEDURE_PARAMETERS System View

Provides information about the stored procedure parameters.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the stored procedure.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the parameter name.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the data type ID.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the data type name.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the parameter length.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Displays the scale of the parameter.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the parameter.</td>
</tr>
<tr>
<td>TABLE_TYPE_SCHEMA</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the table type if DATA_TYPE_NAME is TABLE_TYPE.</td>
</tr>
<tr>
<td>TABLE_TYPE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table type if DATA_TYPE_NAME is TABLE_TYPE.</td>
</tr>
<tr>
<td>IS_INPLACE_TYPE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the parameter type is an inplace type: TRUE/FALSE.</td>
</tr>
<tr>
<td>PARAMETER_TYPE</td>
<td>VARCHAR(7)</td>
<td>Displays the parameter mode: IN, OUT, or INOUT.</td>
</tr>
<tr>
<td>HAS_DEFAULT_VALUE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the parameter has a default value: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the parameter accepts a NULL value: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
Related Information

Procedure Parameters
PROCEDURE_PARAMETER_COLUMNS System View [page 1616]
PROCEDURES System View [page 1612]
PROCEDURE_OBJECTS System View [page 1614]
PROCEDURE_ROUTES System View [page 1617]
Configuration Parameters

6.1.115 PROCEDURE_PARAMETER_COLUMNS System View

Lists available columns of table parameters of stored procedures.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the same of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the stored procedure.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the parameter name.</td>
</tr>
<tr>
<td>PARAMETER_POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the parameter.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column in the table parameter.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the column in the table parameter.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the SQL data type name of the column.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the number of characters for character types, the number of max</td>
</tr>
<tr>
<td></td>
<td></td>
<td>digits for numeric types, the number of characters for datetime types, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the number of bytes for LOB types.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>For numeric types, displays the maximum number of digits to the right of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the decimal point. For time and timestamp types, displays the decimal digits</td>
</tr>
<tr>
<td></td>
<td></td>
<td>defined as the number of digits to the right of the decimal point in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>second's component of the data.</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is allowed to accept NULL values: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

### Related Information

- PROCEDURES System View [page 1612]
- PROCEDURE_OBJECTS System View [page 1614]
- PROCEDURE_PARAMETERS System View [page 1615]
- PROCEDURE_ROUTES System View [page 1617]
- Column View Parameter Binding
- Procedure Parameters
- Array Parameters for Procedures and Functions

### 6.1.116 PROCEDURE_ROUTES System View

Provides information about the procedure being routed. This view is for internal use only.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCEDURE_SCHEMA</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the stored procedure.</td>
</tr>
</tbody>
</table>
## Column name

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the stored procedure.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(13)</td>
<td>Displays the object type: REMOTE_SOURCE, VOLUME, or TABLE.</td>
</tr>
<tr>
<td>OBJECT_SCHEMA</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema of the object, if applicable.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID.</td>
</tr>
</tbody>
</table>

## Related Information

- PROCEDES System View [page 1612]
- PROCEDURE_OBJECTS System View [page 1614]
- PROCEDURE_PARAMETERS System View [page 1615]
- PROCEDURE_PARAMETER_COLUMNS System View [page 1616]
- Procedures

## 6.1.117 PROJECTION_VIEW_COLUMN_SOURCES System View

Provides information about available projection view columns.

## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view column name.</td>
</tr>
<tr>
<td>SOURCE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the source table schema name.</td>
</tr>
<tr>
<td>SOURCE_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the source table name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SOURCE_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the source column name.</td>
</tr>
</tbody>
</table>

Related Information

CREATE PROJECTION VIEW Statement (Data Definition) [page 794]
VIEW_COLUMNS System View [page 1705]

6.1.118 PSES System View

Provides information about personal security environments (PSE).

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSE_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of the PSE.</td>
</tr>
<tr>
<td>NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the PSE.</td>
</tr>
<tr>
<td>PURPOSE</td>
<td>VARCHAR(16)</td>
<td>Displays the purpose of the PSE.</td>
</tr>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the owner of the PSE.</td>
</tr>
</tbody>
</table>

Related Information

PSE_CERTIFICATES System View [page 1620]
PSE_PURPOSE_OBJECTS System View [page 1621]
CREATE PSE Statement (System Management) [page 797]
SET PSE Statement (System Management) [page 1167]
UNSET PSE Statement (System Management) [page 1181]
ALTER PSE Statement (System Management) [page 468]
DROP PSE Statement (System Management) [page 964]
Certificate Collections
Certificate Management in SAP HANA
SQL Statements and Authorization for In-Database Certificate Management (Reference)
Securing Data Communication
6.1.119 PSE_CERTIFICATES System View

Provides information about certificates used in PSEs.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the PSE.</td>
</tr>
<tr>
<td>CERTIFICATE_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of the certificate.</td>
</tr>
<tr>
<td>SUBJECT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the subject of the X.509 certificate.</td>
</tr>
<tr>
<td>ISSUER_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the issuer of the X.509 certificate.</td>
</tr>
<tr>
<td>VALID_FROM</td>
<td>TIMESTAMP</td>
<td>Displays the start time of certificate's validity.</td>
</tr>
<tr>
<td>VALID_UNTIL</td>
<td>TIMESTAMP</td>
<td>Displays the end time of certificate's validity.</td>
</tr>
<tr>
<td>CERTIFICATE_USAGE</td>
<td>VARCHAR(5)</td>
<td>Displays the certificate usage: OWN, CHAIN, or TRUST.</td>
</tr>
<tr>
<td>CERTIFICATE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the unique name of the certificate.</td>
</tr>
</tbody>
</table>

Related Information

PSES System View [page 1619]
PSE_PURPOSE_OBJECTS System View [page 1621]
CREATE PSE Statement (System Management) [page 797]
SET PSE Statement (System Management) [page 1167]
UNSET PSE Statement (System Management) [page 1181]
ALTER PSE Statement (System Management) [page 468]
DROP PSE Statement (System Management) [page 964]
Certificate Collections
Certificate Management in SAP HANA
SQL Statements and Authorization for In-Database Certificate Management (Reference)
Securing Data Communication
6.1.120 PSE_PUBLIC.Keys System View

Provides information about all public keys in a PSE store.

Structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the PSE.</td>
</tr>
<tr>
<td>PUBLIC_KEY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the unique name of the public key.</td>
</tr>
</tbody>
</table>

Permissions

For information on the permissions required to use this view, see SQL Statements and Authorization for Certificate Management (Reference).

Related Information

SQL Statements and Authorization for In-Database Certificate Management (Reference)
PUBLIC.Keys System View [page 1623]

6.1.121 PSE_PURPOSE.OBJECTS System View

Provides information about all PSEs and their assigned providers or hosts, referred to as purpose objects.

Structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Purpose</th>
<th>Purpose Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the PSE.</td>
</tr>
<tr>
<td>PURPOSE</td>
<td>NVARCHAR(24)</td>
<td>Displays the purpose of the PSE store.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Purpose</td>
<td>Purpose Object</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PURPOSE_OBJECT</td>
<td>NVARCHAR(256)</td>
<td>For JWT or SAML purposes, displays the name of the assigned provider. For the SSL purpose, this is the assigned host name.</td>
</tr>
</tbody>
</table>

**Additional Information**

One PSE may have multiple lines, one line per assigned provider or host. All PSES are shown, including those with and without a purpose and those with and without a purpose object assigned.

**Permissions**

No additional privilege is required to see those entries for which you have the REFERENCES or ALTER privilege on the PSE object. To see all entries in the PSE_PURPOSE_OBJECTS system view, you must have the TRUST ADMIN system privilege.

**Related Information**

- PSES System View [page 1619]
- PSE_CERTIFICATES System View [page 1620]
- CREATE PSE Statement (System Management) [page 797]
- SET PSE Statement (System Management) [page 1167]
- UNSET PSE Statement (System Management) [page 1181]
- ALTER PSE Statement (System Management) [page 468]
- DROP PSE Statement (System Management) [page 964]
- Certificate Collections
- Certificate Management in SAP HANA
- SQL Statements and Authorization for In-Database Certificate Management (Reference)
- Securing Data Communication
## 6.1.122 PUBLIC_KEYS System View

Provides information about all public keys.

### Structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUBLIC_KEY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays a unique name of the public key</td>
</tr>
<tr>
<td>ALGORITHM</td>
<td>CHAR(3)</td>
<td>Displays the algorithm of this key (RSA or EC)</td>
</tr>
<tr>
<td>ALGORITHM_DETAILS</td>
<td>NVARCHAR(64)</td>
<td>Displays the keysize for RSA or the used curve for EC</td>
</tr>
<tr>
<td>KEY</td>
<td>NVARCHAR(5000)</td>
<td>Displays the PEM representation of the key (PKCS1 or PKCS8)</td>
</tr>
<tr>
<td>KEY_ID_HINT</td>
<td>NVARCHAR(256)</td>
<td>Displays the hint ID this key has in an external key management system (e.g. JWKS)</td>
</tr>
<tr>
<td>COMMENT</td>
<td>NVARCHAR(1024)</td>
<td>Displays the description for the key</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation timestamp of the public key in the database</td>
</tr>
</tbody>
</table>

### Permissions

For information on the permissions required to use this view, see *SQL Statements and Authorization for Certificate Management (Reference)*.

### Related Information

- *SQL Statements and Authorization for In-Database Certificate Management (Reference)*
- PSE_PURPOSE_OBJECTS System View [page 1621]
6.1.123 QUERY_PLANS System View

Plans how to handle query execution.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAN_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the plan ID.</td>
</tr>
<tr>
<td>OPERATOR_NAME</td>
<td>VARCHAR(5000)</td>
<td>Displays the operator name.</td>
</tr>
<tr>
<td>OPERATOR_DETAILS</td>
<td>NCLOB</td>
<td>Displays the detailed information on operators in the query plan.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>SUBTREE_COST</td>
<td>DOUBLE</td>
<td>Displays the subtree cost.</td>
</tr>
<tr>
<td>INPUT_CARDINALITY</td>
<td>DOUBLE</td>
<td>Displays the input cardinality.</td>
</tr>
<tr>
<td>OUTPUT_CARDINALITY</td>
<td>DOUBLE</td>
<td>Displays the output cardinality.</td>
</tr>
<tr>
<td>OPERATOR_ID</td>
<td>INTEGER</td>
<td>Displays the operator ID.</td>
</tr>
<tr>
<td>PARENT_OPERATOR_ID</td>
<td>INTEGER</td>
<td>Displays the parent operator ID.</td>
</tr>
<tr>
<td>LEVEL</td>
<td>INTEGER</td>
<td>Displays the level.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the position.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(256)</td>
<td>Displays the host where the plan operator is executed.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port where the plan operator is executed.</td>
</tr>
</tbody>
</table>

**Related Information**

Query Parameterization: BIND_AS_PARAMETER and BIND_AS_VALUE
Query Export
SQLScript Query Export

SAP HANA SQL Reference Guide for SAP HANA Platform
System Views Reference
### 6.1.124 REFERENTIAL_CONSTRAINTS System View

Provides information about referential constraints.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>POSITION</td>
<td>SMALLINT</td>
<td>Displays the column position in this constraint.</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the constraint name.</td>
</tr>
<tr>
<td>REFERENCED_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the table referenced by this constraint.</td>
</tr>
<tr>
<td>REFERENCED_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table referenced by this constraint.</td>
</tr>
<tr>
<td>REFERENCED_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column referenced by this column.</td>
</tr>
<tr>
<td>REFERENCED_CONSTRAINT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the unique constraint referenced by this constraint.</td>
</tr>
<tr>
<td>UPDATE_RULE</td>
<td>VARCHAR(16)</td>
<td>Displays the update rule: CASCADE, SET NULL, SET DEFAULT, or RESTRICT.</td>
</tr>
<tr>
<td>DELETE_RULE</td>
<td>VARCHAR(16)</td>
<td>Displays the delete rule: CASCADE, SET NULL, SET DEFAULT, or RESTRICT.</td>
</tr>
<tr>
<td>IS_ENFORCED</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not this constraint is enforced: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_VALIDATED</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not this constraint is validated: TRUE/FALSE.</td>
</tr>
<tr>
<td>CHECK_TIME</td>
<td>STRING</td>
<td>Displays the time when the constraint is checked: INITIALLY_IMMEDIATE/INITIALLY_DEFERRED.</td>
</tr>
</tbody>
</table>


Related Information

CONSTRAINTS System View [page 1523]
CREATE TABLE Statement (Data Definition) [page 825]
ALTER TABLE Statement (Data Definition) [page 589]
CS_JOIN_CONSTRAINTS System View [page 1530]
Time Selection Partitioning (Aging)
SQL DML Statements on Table Variables
Table Variable Type Definition

6.1.125 REMOTE_SOURCES System View

Provides information about remote sources.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBSCRIPTION_OID</td>
<td>BIGINT</td>
<td>Displays the remote subscription OID.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the adapter name.</td>
</tr>
<tr>
<td>CONNECTION_INFO</td>
<td>NVARCHAR(256)</td>
<td>Displays the connection information.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>VARCHAR(11)</td>
<td>Displays the adapter location.</td>
</tr>
<tr>
<td>AGENT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the agent name.</td>
</tr>
<tr>
<td>IS_CDC_SUPPORTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the CDC is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_REFRESH_OBJECTS_SUPPORTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not refresh objects is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>AGENT_GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the agent group name.</td>
</tr>
<tr>
<td>IS_TREE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the browsing hierarchy is deeply nested: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
### RELATED INFORMATION

**RELATED INFORMATION**

**REMOTE_SOURCE_OBJECTS System View [page 1627]**
**REMOTE_SOURCE_OBJECT_COLUMNS System View [page 1629]**
**REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINTS System View [page 1630]**
**REMOTE_SOURCE_OBJECT_COLUMN_DESCRIPTIONS System View [page 1631]**
**REMOTE_SOURCE_OBJECT_DESCRIPTIONS System View [page 1632]**
**CREATE REMOTE SOURCE Statement (Access Control) [page 799]**
**ALTER REMOTE SOURCE Statement (Access Control) [page 471]**
**DROP REMOTE SOURCE Statement (Access Control) [page 966]**

**Creating Remote Sources**
**Modify a Remote Source**
**Drop a Remote Source**
**List Remote Sources**
**Customizing the Behavior of a Remote Source**

### RELATED INFORMATION

**REMOTE_SOURCE_OBJECTS System View [page 1627]**
**REMOTE_SOURCE_OBJECT_COLUMNS System View [page 1629]**
**REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINTS System View [page 1630]**
**REMOTE_SOURCE_OBJECT_COLUMN_DESCRIPTIONS System View [page 1631]**
**REMOTE_SOURCE_OBJECT_DESCRIPTIONS System View [page 1632]**
**CREATE REMOTE SOURCE Statement (Access Control) [page 799]**
**ALTER REMOTE SOURCE Statement (Access Control) [page 471]**
**DROP REMOTE SOURCE Statement (Access Control) [page 966]**

**Creating Remote Sources**
**Modify a Remote Source**
**Drop a Remote Source**
**List Remote Sources**
**Customizing the Behavior of a Remote Source**

### 6.1.126 REMOTE_SOURCE_OBJECTS System View

Provides remote source object information.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the unique name to identify the remote source object.</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name for the object.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_IMPORTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the object is importable as a virtual table: TRUE/</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE.</td>
</tr>
<tr>
<td>IS_EXPANDABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the object can be expanded or browsed to get inner</td>
</tr>
<tr>
<td></td>
<td></td>
<td>objects: TRUE/FALSE.</td>
</tr>
<tr>
<td>PARENT_OBJECT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the parent object name for the specified object.</td>
</tr>
<tr>
<td>DEFINITION_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object definition type.</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NCLOB</td>
<td>Displays the object definition.</td>
</tr>
</tbody>
</table>

**Related Information**

REMOTE_SOURCES System View [page 1626]
REMOTE_SOURCE_OBJECT_COLUMNS System View [page 1629]
REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINTS System View [page 1630]
REMOTE_SOURCE_OBJECT_COLUMN_DESCRIPTIONS System View [page 1631]
REMOTE_SOURCE_OBJECT_DESCRIPTIONS System View [page 1632]
REMOTE_SUBSCRIPTIONS System View [page 1633]
REMOTE_SUBSCRIPTION_DATA_CONTAINERS System View [page 1635]
CREATE REMOTE SOURCE Statement (Access Control) [page 799]
ALTER REMOTE SOURCE Statement (Access Control) [page 471]
DROP REMOTE SOURCE Statement (Access Control) [page 966]
Creating Remote Sources
Modify a Remote Source
Drop a Remote Source
List Remote Sources
Customizing the Behavior of a Remote Source
6.1.127 REMOTE_SOURCE_OBJECT_COLUMNS System View

Provides remote source object column information.

**Structure**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the unique name to identify the remote source object.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the SAP HANA data type.</td>
</tr>
<tr>
<td>REMOTE_DATA_TYPE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote data type.</td>
</tr>
<tr>
<td>REMOTECONTENT_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote content type.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the length of the column. If there is a decimal then this value is the precision.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Displays the scale.</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not this column is allowed to be NULL: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_AUTOINCREMENT</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not this column is automatically incremented: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

REMOTE_SOURCES System View [page 1626]
REMOTE_SOURCE_OBJECTS System View [page 1627]
REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINTS System View [page 1630]
REMOTE_SOURCE_OBJECT_COLUMN_DESCRIPTIONS System View [page 1631]
REMOTE_SOURCE_OBJECT_DESCRIPTIONS System View [page 1632]
REMOTE_SUBSCRIPTIONS System View [page 1633]
REMOTE_SUBSCRIPTION_DATA_CONTAINERS System View [page 1635]
6.1.128  REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINT
S System View

Provides remote source object column constraint information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the unique name to identify the remote source object.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the position of the column.</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the constraint name.</td>
</tr>
<tr>
<td>IS_PRIMARY_KEY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the specified key is the primary key: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_UNIQUE_KEY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the specified key is unique: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

REMOTE_SOURCES System View [page 1626]
REMOTE_SOURCE_OBJECTS System View [page 1627]
REMOTE_SOURCE_OBJECT_COLUMNS System View [page 1629]
6.1.129 REMOTE_SOURCE_OBJECT_COLUMN_DESCRIPTIONS System View

Provides remote source object column description information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the unique name to identify the remote source object.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column.</td>
</tr>
<tr>
<td>LANGUAGE_CODE</td>
<td>VARCHAR(2)</td>
<td>Displays the language code.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the description of the specified column.</td>
</tr>
</tbody>
</table>

**Related Information**

REMOTE_SOURCES System View [page 1626]
REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINTS System View [page 1630]
REMOTE_SOURCE_OBJECT_COLUMNS System View [page 1629]
6.1.130  REMOTE_SOURCE_OBJECT_DESCRIPTIONS System View

Provides remote source object description information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the unique name to identify the remote source object.</td>
</tr>
<tr>
<td>LANGUAGE_CODE</td>
<td>VARCHAR(2)</td>
<td>Displays the language code.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the description of the specified object.</td>
</tr>
</tbody>
</table>

Related Information

REMOTE_SOURCES System View [page 1626]
REMOTE_SOURCE_OBJECTS System View [page 1627]
REMOTE_SOURCE_OBJECT_COLUMNS System View [page 1629]
REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINTS System View [page 1630]
REMOTE_SOURCE_OBJECT_COLUMN_DESCRIPTIONS System View [page 1631]
6.1.131 REMOTE_SUBSCRIPTIONS System View

Lists all the remote subscriptions created for a remote source.

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBSCRIPTION_OID</td>
<td>BIGINT</td>
<td>Displays the remote subscription OID.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote subscription schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote subscription name.</td>
</tr>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the owner’s name.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the remote subscription is valid or not: TRUE/FALSE. This value is FALSE when its source or target objects are changed or dropped.</td>
</tr>
<tr>
<td>SUBSCRIPTION_TYPE</td>
<td>VARCHAR(13)</td>
<td>Displays the remote subscription type.</td>
</tr>
<tr>
<td>VIRTUAL_TABLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the virtual table schema name.</td>
</tr>
<tr>
<td>VIRTUAL_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the virtual table name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_QUERY_STRING</td>
<td>NCLOB</td>
<td>Displays the SELECT statement specified in the subscription when subscription type is SQL.</td>
</tr>
<tr>
<td>TARGET_OBJECT_TYPE</td>
<td>VARCHAR(9)</td>
<td>Displays the remote subscription target object type: TABLE, PROCEDURE, or TASK.</td>
</tr>
<tr>
<td>TARGET_OBJECT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the target object schema name.</td>
</tr>
<tr>
<td>TARGET_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the target object name.</td>
</tr>
<tr>
<td>Column</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TARGET_OTHER_PARAM_STRING</td>
<td>NVARCHAR(4000)</td>
<td>Displays the constant parameter string to pass during execution when the target object type is PROCEDURE or TASK.</td>
</tr>
<tr>
<td>CHANGE_MODE</td>
<td>VARCHAR(6)</td>
<td>Displays the remote subscription change mode: NULL, NORMAL, UPSERT, or INSERT.</td>
</tr>
<tr>
<td>CHANGE_TYPE_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote subscription change type column name in the target table: I (INSERT), D (DELETE), and so on.</td>
</tr>
<tr>
<td>CHANGE_TIME_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote subscription change time column name in the target table. Supported data types are TIMESTAMP, CHAR, and VARCHAR with a minimum length of 27.</td>
</tr>
<tr>
<td>CHANGE_SEQUENCE_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote subscription change sequence column name in the target table.</td>
</tr>
<tr>
<td>SCHEMA_CHANGES</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the remote subscription propagates schema changes: TRUE/FALSE.</td>
</tr>
<tr>
<td>TASK_PROCEDURE_PARAMETERS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the comma-separated list of task parameters.</td>
</tr>
</tbody>
</table>

**Related Information**

REMOTE_SUBSCRIPTION_DATA_CONTAINERS System View [page 1635]
REMOTE_SUBSCRIPTION_EXCEPTIONS System View [page 1636]
REMOTE_USERS System View [page 1637]
REMOTE_SOURCES System View [page 1626]
CREATE REMOTE SUBSCRIPTION Statement [Smart Data Integration]
ALTER REMOTE SUBSCRIPTION Statement [Smart Data Integration]
DROP REMOTE SUBSCRIPTION Statement [Smart Data Integration]
6.1.132 REMOTE_SUBSCRIPTION_DATA_CONTAINERS
System View

Provides information regarding remote subscription data.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>CONTENT_TYPE</td>
<td>VARCHAR(40)</td>
<td>Displays the category of data containers:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• COMMIT SEQUENCE GROUP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• COMMIT SEQUENCES (displays the order that transactions get committed and a sequence number for each commit row)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TRANSACTION (each container contains only one transaction)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LOBS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ROLLBACK SAVEPOINTS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DISTRIBUTOR QUEUE DATA (displays the data currently being queued by the distributor)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DISTRIBUTOR QUEUE DATA REFERENCES (displays the list of transactions currently being referenced by the distributor)</td>
</tr>
<tr>
<td>SUBSCRIPTION_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the remote subscription.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the remote subscription.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>VARCHAR(512)</td>
<td>Displays the ID of the transaction being replicated.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the data container. Depending on the content type, a container can be searched by REMOTE_SOURCE_ID and CONTENT_TYPE or with additional information such as SUBSCRIPTION_OID or TRANSACTION_ID.</td>
</tr>
</tbody>
</table>
6.1.133 REMOTE_SUBSCRIPTION_EXCEPTIONS System View

Provides remote subscription exception information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCEPTION_OID</td>
<td>BIGINT</td>
<td>Displays the exception ID.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(19)</td>
<td>Displays the object type: REMOTE_SOURCE/REMOTE_SUBSCRIPTION.</td>
</tr>
<tr>
<td>OBJECT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the remote source or the remote subscription, based on the OBJECT_TYPE.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of the remote source or remote subscription, based on the OBJECT_TYPE.</td>
</tr>
<tr>
<td>EXCEPTION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time at which the exception was raised.</td>
</tr>
<tr>
<td>ERROR_NUMBER</td>
<td>INTEGER</td>
<td>Displays the error number.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VARCHAR(11)</td>
<td>Displays the component that raised the exception.</td>
</tr>
</tbody>
</table>
6.1.134 REMOTE_USERS System View

Provides information about user mappings for cross-database access.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the name of the user in the local database.</td>
</tr>
<tr>
<td>REMOTE_USER_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the name of the user in the remote database.</td>
</tr>
<tr>
<td>REMOTE_DATABASE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the name of the remote database.</td>
</tr>
</tbody>
</table>

Related Information

REMOTE_SOURCES System View [page 1626]
REMOTE_SUBSCRIPTIONS System View [page 1633]
REMOTE_SUBSCRIPTION_DATA_CONTAINERS System View [page 1635]
REMOTE_USERS System View [page 1637]
CREATE REMOTE SUBSCRIPTION Statement [Smart Data Integration]
ALTER REMOTE SUBSCRIPTION Statement [Smart Data Integration]
DROP REMOTE SUBSCRIPTION Statement [Smart Data Integration]
6.1.135 REORG_GENERATE_OVERVIEW System View

Tracks automated and administrator calls to the REORG_GENERATE procedure.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERATE_ID</td>
<td>BIGINT</td>
<td>Displays the generated ID. This is an increasing, unique number used to identify the call to the REORG_GENERATE procedure.</td>
</tr>
<tr>
<td>START_DATE</td>
<td>TIMESTAMP</td>
<td>Displays the UTC timestamp when the call reached the engine.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID that initiates the call to the REORG_GENERATE procedure.</td>
</tr>
<tr>
<td>USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>ALGORITHM_ID</td>
<td>INTEGER</td>
<td>Displays the table redistribution algorithm ID used in the call.</td>
</tr>
<tr>
<td>REORG_ID</td>
<td>INTEGER</td>
<td>Displays the reorganization ID used to select the corresponding executed table redistribution plan.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>NVARCHAR(4096)</td>
<td>Displays the table redistribution parameters that are used in the call.</td>
</tr>
<tr>
<td>DETAIL</td>
<td>NVARCHAR(1024)</td>
<td>Displays the detailed text for additional information.</td>
</tr>
</tbody>
</table>

Related Information

REORG_OVERVIEW System View [page 1639]
REORG_PLAN System View [page 1640]
REORG_PLAN_INFOS System View [page 1641]
REORG_STEPS System View [page 1642]
M_REORG_ALGORITHMS System View [page 2083]
PROCEDURES System View [page 1612]
6.1.136 REORG_OVERVIEW System View

Provides an overview of landscape redistributions.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORG_ID</td>
<td>INTEGER</td>
<td>Displays the unique identifier for the executed reorg plan.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the status of the plan execution:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STARTED: The plan execution has started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• FINISHED: The plan execution finished successfully.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• FAILED: The plan execution is treated as failed.</td>
</tr>
<tr>
<td>START_DATE</td>
<td>TIMESTAMP</td>
<td>Displays the UTC timestamp when the plan execution started.</td>
</tr>
<tr>
<td>END_DATE</td>
<td>TIMESTAMP</td>
<td>Displays the UTC timestamp when plan execution stopped. This field remains empty while the execution is running or in the case of reboots during plan execution.</td>
</tr>
<tr>
<td>USER</td>
<td>VARCHAR(256)</td>
<td>Displays the user that executed the table redistribution.</td>
</tr>
<tr>
<td>ALGORITHM_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the algorithm for the table redistribution.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>VARCHAR(1024)</td>
<td>Displays the parameters given for the table redistribution call.</td>
</tr>
</tbody>
</table>

Additional Information

This view contains information about table redistribution executions on this system and stores every call to the REORG_EXECUTE statement. You can use this view to determine execution durations and the status of each table redistribution on this system. Only executed plan information is stored and calls of REORG_GENERATE without REORG_EXECUTE are not displayed.
Related Information

REORG_GENERATE_OVERVIEW System View [page 1638]
REORG_PLAN System View [page 1640]
REORG_PLAN_INFOS System View [page 1641]
REORG_STEPS System View [page 1642]
M_REORG_ALGORITHMS System View [page 2083]

6.1.137 REORG_PLAN System View

Provides current plan information for landscape reorganization.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP_ID</td>
<td>INTEGER</td>
<td>Displays the step ID.</td>
</tr>
<tr>
<td>STEP_GROUP</td>
<td>INTEGER</td>
<td>Displays the step group.</td>
</tr>
<tr>
<td>PRECONDITION</td>
<td>VARCHAR(1024)</td>
<td>Displays the precondition of the specified step.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>NEW_PARTITION_SPEC</td>
<td>NCLOB</td>
<td>Displays the new partition specification.</td>
</tr>
<tr>
<td>OLD_PARTITION_SPEC</td>
<td>NCLOB</td>
<td>Displays the old partition specification.</td>
</tr>
<tr>
<td>PARTITION</td>
<td>INTEGER</td>
<td>Displays the partition ID.</td>
</tr>
<tr>
<td>NEW_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where the table/partition is moved to.</td>
</tr>
<tr>
<td>NEW_PORT</td>
<td>INTEGER</td>
<td>Displays the port where the table/partition is moved to.</td>
</tr>
<tr>
<td>OLD_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where the table/partition is located.</td>
</tr>
<tr>
<td>OLD_PORT</td>
<td>INTEGER</td>
<td>Displays the port where the table/partition is located.</td>
</tr>
</tbody>
</table>
Additional Information

This view contains the last table redistribution plan generated with this database connection. The contents of the session are stored temporarily and are deleted when the connection is closed.

Related Information

REORG_GENERATE_OVERVIEW System View [page 1638]
REORG_OVERVIEW System View [page 1639]
REORG_PLAN_INFOS System View [page 1641]
REORG_STEPS System View [page 1642]
M_REORG_ALGORITHMS System View [page 2083]

6.1.138 REORG_PLAN_INFOS System View

Provides additional information about the current landscape reorganization plan.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY</td>
<td>VARCHAR(256)</td>
<td>Displays the key.</td>
</tr>
<tr>
<td>VALUE</td>
<td>VARCHAR(1024)</td>
<td>Displays the value.</td>
</tr>
</tbody>
</table>

Additional Information

This view contains information about the last table redistribution plan generation with this database connection. The contents are stored for the current session and are deleted when the connection is closed. The following list describes the contents of the individual lines of this view (VALUE), as identified by KEY:

- **MOVE_PHYSICAL**: Indicates that all move operations during plan execution are executed as physical moves. Possible values are 0 or 1.
- **ALGORITHM_ID**: Displays the ID of the reorg algorithm that is used for generating the corresponding table redistribution plan.
- **PARAMETERS**: Displays the parameter information of the last table redistribution plan generation call.
Related Information

REORG_GENERATE_OVERVIEW System View [page 1638]
REORG_OVERVIEW System View [page 1639]
REORG_PLAN System View [page 1640]
REORG_STEPS System View [page 1642]
M_REORG_ALGORITHMS System View [page 2083]

6.1.139 REORG_STEPS System View

Contains the executed or to be executed table redistribution plan items.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORG_ID</td>
<td>INTEGER</td>
<td>Displays the reorg ID.</td>
</tr>
<tr>
<td>STEP_ID</td>
<td>INTEGER</td>
<td>Displays the step ID.</td>
</tr>
<tr>
<td>STEP_GROUP</td>
<td>INTEGER</td>
<td>Displays the ID of the table redistribution group of steps that the item belongs to.</td>
</tr>
<tr>
<td>PRECONDITION</td>
<td>VARCHAR(1024)</td>
<td>Indicates the preconditions that have to be fulfilled before the table redistribution step can be executed.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>NEW_PARTITION_SPEC</td>
<td>NCLOB</td>
<td>Returns the new partition specification.</td>
</tr>
<tr>
<td>OLD_PARTITION_SPEC</td>
<td>NCLOB</td>
<td>Returns the old partition specification.</td>
</tr>
<tr>
<td>PARTITION</td>
<td>INTEGER</td>
<td>Displays the partition ID.</td>
</tr>
<tr>
<td>NEW_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where the table/partition is moved to.</td>
</tr>
<tr>
<td>NEW_PORT</td>
<td>INTEGER</td>
<td>Displays the port where the table/partition is moved to.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OLD_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where the table/partition was located.</td>
</tr>
<tr>
<td>OLD_PORT</td>
<td>INTEGER</td>
<td>Displays the port where the table/partition was located.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the status message.</td>
</tr>
<tr>
<td>ERROR</td>
<td>VARCHAR(1024)</td>
<td>Displays the error message.</td>
</tr>
<tr>
<td>START_DATE</td>
<td>TIMESTAMP</td>
<td>Specifies the UTC timestamp when the plan execution started.</td>
</tr>
<tr>
<td>END_DATE</td>
<td>TIMESTAMP</td>
<td>Specifies the UTC timestamp when the plan execution stopped.</td>
</tr>
</tbody>
</table>

**Related Information**

- REORG_GENERATE_OVERVIEW System View [page 1638]
- REORG_OVERVIEW System View [page 1639]
- REORG_PLAN System View [page 1640]
- REORG_PLAN_INFOS System View [page 1641]
- M_REORG_ALGORITHMS System View [page 2083]

### 6.1.140 RESERVED_KEYWORDS System View

Provides a list of reserved keywords that are not allowed to be used as a simple identifier.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVED_KEYWORD</td>
<td>VARCHAR(256)</td>
<td>Displays the reserved keyword.</td>
</tr>
</tbody>
</table>

**Related Information**

Reserved Words [page 50]
**6.1.141 RESULT_CACHE System View**

 Provides information about objects available to use the result cache.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type: VIEW/FUNCTION.</td>
</tr>
<tr>
<td>OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the object ID.</td>
</tr>
<tr>
<td>CACHE_TYPE</td>
<td>VARCHAR(44)</td>
<td>Displays which type of cache is enabled for the object: FULL, PARTIAL (SELECTED COLUMNS), PARTIAL (FILTERED), or PARTIAL (SELECTED COLUMNS, FILTERED).</td>
</tr>
<tr>
<td>CACHE_RETENTION</td>
<td>INTEGER</td>
<td>Displays the cache refresh interval in minutes.</td>
</tr>
<tr>
<td>CACHE_FILTER</td>
<td>NCLOB</td>
<td>Displays the filter condition specified to reduce the cache size.</td>
</tr>
<tr>
<td>CACHE_LOCATIONS</td>
<td>VARCHAR(5000)</td>
<td>Displays the cache entry locations.</td>
</tr>
<tr>
<td>IS_CACHE_FORCED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the cached is forced or if cachability is checked: TRUE/ FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

- RESULT_CACHE_COLUMN System View [page 1645]
- M_RESULT_CACHE System View [page 2084]
- M_RESULT_CACHE_RESET System View [page 2086]
- M_RESULT_CACHE_EXCLUSIONS System View [page 2086]
- ALTER SYSTEM CLEAR RESULT CACHE Statement (System Management) [page 522]
- ALTER SYSTEM REMOVE RESULT CACHE ENTRY Statement (System Management) [page 561]
- ALTER SYSTEM REFRESH RESULT CACHE ENTRY Statement (System Management) [page 558]
6.1.142 RESULT_CACHE_COLUMNS System View

Provides information about columns available to use the result cache.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>CACHE_AGGREGATION_TYPE</td>
<td>VARCHAR(20)</td>
<td>Indicates the required aggregation type in a query to use cache or indicate an aggregation type for caching the column: SUM, MIN, MAX, or COUNT.</td>
</tr>
</tbody>
</table>

Related Information

RESULT_CACHE System View [page 1644]
M_RESULT_CACHE System View [page 2084]
M_RESULT_CACHE_RESET System View [page 2086]
M_RESULT_CACHE_EXCLUSIONS System View [page 2086]
ALTER SYSTEM CLEAR RESULT CACHE Statement (System Management) [page 522]
ALTER_SYSTEM_REMOVE_RESULT_CACHE ENTRY Statement (System Management) [page 561]
ALTER SYSTEM REFRESH RESULT CACHE ENTRY Statement (System Management) [page 558]
ALTER SYSTEM REFRESH RESULT CACHE Statement (System Management) [page 557]
Results Caching for Virtual Tables and Linked Database
Procedure Result Cache
Deterministic Procedure Cache
## 6.1.143 ROLES System View

Shows available roles.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema of the role granted.</td>
</tr>
<tr>
<td>ROLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the role name.</td>
</tr>
<tr>
<td>ROLE_ID</td>
<td>BIGINT</td>
<td>Displays the role ID.</td>
</tr>
<tr>
<td>ROLE_MODE</td>
<td>VARCHAR(5)</td>
<td>Displays the local mode of the role.</td>
</tr>
<tr>
<td>GLOBAL_IDENTITY</td>
<td>NVARCHAR(256)</td>
<td>Displays the identity specified for the role with ROLE_MODE GLOBAL.</td>
</tr>
<tr>
<td>CREATOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user who created the role.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the role was created.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the description for the specified role.</td>
</tr>
<tr>
<td>CONTEXT</td>
<td>NVARCHAR(2048)</td>
<td>Displays the context that the role is valid for.</td>
</tr>
</tbody>
</table>

### Related Information

- CREATE ROLE Statement (Access Control) [page 802]
- ALTER ROLE Statement (Access Control) [page 475]
- REVOKE Statement (Access Control) [page 1098]
- DROP ROLE Statement (Access Control) [page 967]
- ROLE_LDAP_GROUPS System View [page 1647]
- GRANTED_ROLES System View [page 1581]
- EFFECTIVE_ROLES System View [page 1553]
- EFFECTIVE_ROLE_GRANTEES System View [page 1554]
- Database Roles
- Change a Role
Delete a Role
Database Role Details
Organizational Roles
Space Roles
Assign Roles to Database Users
Process Engine Roles
Create a Catalog Role
Configuring Host Roles
User Self-Service Roles
Adding Host Roles
Removing Host Roles
Prerequisites for Granting and Revoking Privileges and Roles

6.1.144 ROLE_LDAP_GROUPS System View

Shows the mapping between SAP HANA roles and LDAP groups.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the role.</td>
</tr>
<tr>
<td>ROLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the role.</td>
</tr>
<tr>
<td>LDAP_GROUP_NAME</td>
<td>NVARCHAR(1024)</td>
<td>Displays the name of the LDAP group.</td>
</tr>
</tbody>
</table>

Related Information

CREATE LDAP PROVIDER Statement (Access Control) [page 772]
ALTER LDAP PROVIDER Statement (Access Control) [page 458]
VALIDATE LDAP PROVIDER Statement (Access Control) [page 1195]
DROP LDAP PROVIDER Statement (Access Control) [page 962]
LDAP_USERS System View [page 1599]
LDAP_PROVIDERS System View [page 1597]
CREATE ROLE Statement (Access Control) [page 802]
ALTER ROLE Statement (Access Control) [page 475]
DROP ROLE Statement (Access Control) [page 967]
6.1.145 SAML_PROVIDERS System View

Shows available SAML providers.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAML_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the provider name.</td>
</tr>
<tr>
<td>SUBJECT_NAME</td>
<td>NVARCHAR(512)</td>
<td>Displays the subject of the X.509 certificate of the SAML provider.</td>
</tr>
<tr>
<td>ISSUER_NAME</td>
<td>NVARCHAR(512)</td>
<td>Displays the issuer of the X.509 certificate of the SAML provider.</td>
</tr>
<tr>
<td>ENTITY_ID</td>
<td>NVARCHAR(1024)</td>
<td>Displays the entity name of the SAML provider (optional).</td>
</tr>
<tr>
<td>IS_USER_CREATION_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the SAML provider is allowed to create a user implicitly: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_CASE_SENSITIVE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the user mapping is checked case sensitive: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_USER_CREATION_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the provider is allowed to create a user implicitly: TRUE/ FALSE.</td>
</tr>
<tr>
<td>USER_CREATION_USER_TYPE</td>
<td>VARCHAR(10)</td>
<td>Displays the type of user to create when user creation is enabled: STANDARD, RESTRICTED</td>
</tr>
<tr>
<td>USER_CREATION_USERGROUP</td>
<td>NVARCHAR(256)</td>
<td>Displays the user group a created user will be a member of.</td>
</tr>
<tr>
<td>USER_CREATION_AUTHORIZATION_MODE</td>
<td>NVARCHAR(5)</td>
<td>Displays the authorization mode of the automatic user creation. Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• NULL (automatic user creation is off)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LDAP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LOCAL</td>
</tr>
</tbody>
</table>
Related Information

CREATE SAML PROVIDER Statement (Access Control) [page 803]
ALTER SAML PROVIDER Statement (Access Control) [page 476]
DROP SAML PROVIDER Statement (Access Control) [page 969]
Maintaining SAML Providers
Configure an SAP HANA System as an SAML Service Provider
SAML Service Provider Details
Add an SAML Identity Provider
SAML Identity Provider Details
Modify an Existing SAML Identity Provider

6.1.146 SAML_USER_MAPPINGS System View

Shows the SAML providers known for each user.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>SAML_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the provider name.</td>
</tr>
<tr>
<td>EXTERNAL.IDENTITY</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user known to the provider.</td>
</tr>
</tbody>
</table>

Related Information

SAML_PROVIDERS System View [page 1648]
CREATE SAML PROVIDER Statement (Access Control) [page 803]
ALTER SAML PROVIDER Statement (Access Control) [page 476]
DROP SAML PROVIDER Statement (Access Control) [page 969]
Maintaining SAML Providers
Configure an SAP HANA System as an SAML Service Provider
SAML Service Provider Details
Add an SAML Identity Provider
SAML Identity Provider Details
Modify an Existing SAML Identity Provider
### 6.1.147 SCHEDULER_JOB_PARAMETERS System View

Shows parameters for the jobs.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the job.</td>
</tr>
<tr>
<td>SCHEDULER_JOB_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the job.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the parameter.</td>
</tr>
<tr>
<td>PARAMETER_VALUE</td>
<td>NCLOB</td>
<td>Displays the parameter value as a Unicode string.</td>
</tr>
</tbody>
</table>

**Additional Information**

Users see jobs that they created or on which they have been granted the ALTER or DROP object privilege. Users with the CATALOG READ system privilege see parameters for all jobs.

### 6.1.148 SCHEDULER_JOBS System View

Shows existing jobs.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user who owns the job. The job is executed as this user.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the job.</td>
</tr>
<tr>
<td>SCHEDULER_JOB_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the job.</td>
</tr>
<tr>
<td>PROCEDURE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the called procedure.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the called procedure.</td>
</tr>
<tr>
<td>CRON</td>
<td>NVARCHAR(256)</td>
<td>Displays the schedule of the job.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>If specified, displays the earliest time after which the scheduled job can start to run; otherwise null.</td>
</tr>
</tbody>
</table>
### Column Names

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>If specified, displays the latest time before which the scheduled job can start to run; otherwise null.</td>
</tr>
<tr>
<td>IS_ENABLED</td>
<td>NVARCHAR(5)</td>
<td>Displays whether the scheduled job can run: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>NVARCHAR(5)</td>
<td>Displays whether the job is valid: TRUE/FALSE. If the procedure becomes invalid or if the owner of the job no longer has the EXECUTE privilege for the procedure, invalid jobs are not scheduled.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays any comment on the job.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time at which the job was created.</td>
</tr>
</tbody>
</table>

### Additional Information

Users see jobs that they created or on which they have been granted the ALTER or DROP object privilege. Users with the CATALOG READ system privilege see all jobs.

### 6.1.149 SCHEMAS System View

Shows available schemas.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>SCHEMA_OWNER</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema owner.</td>
</tr>
<tr>
<td>HAS_PRIVILEGES</td>
<td>VARCHAR(5)</td>
<td>Shows if the user is the schema owner or has any privileges for the schema or any object within the schema: TRUE/FALSE.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time of the schema.</td>
</tr>
</tbody>
</table>
Related Information

CREATE SCHEMA Statement (Data Definition) [page 807]
SET SCHEMA Statement (Session Management) [page 1170]
RENAME SCHEMA Statement (Data Definition) [page 1092]
DROP SCHEMA Statement (Data Definition) [page 971]
CREATE SCHEMA SYNONYM Statement (Data Definition) [page 808]
DROP SCHEMA SYNONYM Statement (Data Definition) [page 972]
CURRENT_OBJECT_SCHEMA Function (Miscellaneous) [page 169]
List Virtual Tables By Schema
Import Schemas, Tables, and Other Catalog Objects
Export Schemas, Tables, and Other Catalog Objects

6.1.150  SEARCH_RULE_SETS System View

Shows information about available search rule sets.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEARCH_RULE_SET_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the runtime schema of the search rule set.</td>
</tr>
<tr>
<td>SEARCH_RULE_SET_PACKAGE_ID</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the repository package of the search rule set.</td>
</tr>
<tr>
<td>SEARCH_RULE_SET_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the search rule set.</td>
</tr>
<tr>
<td>SEARCHED_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the catalog object that is searched by the given search rule set.</td>
</tr>
<tr>
<td>SEARCHED_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the catalog object that is searched by the given search rule set.</td>
</tr>
<tr>
<td>SEARCHED_REPOSITORY_PACKAGE_ID</td>
<td>NVARCHAR(256)</td>
<td>Displays the ID of the repository package of the repository object that is searched by the given search rule set.</td>
</tr>
</tbody>
</table>
### Related Information

**SEARCH_RULE_SET_CONDITIONS System View** ([page 1653](#))

#### Search, Sort, and Filter Buttons

- **Search in Table Variables**

### 6.1.151 SEARCH_RULE_SET_CONDITIONS System View

Shows conditions to available search rule sets.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEARCHED_REPOSITORY_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the repository object that is searched by the given search rule set.</td>
</tr>
<tr>
<td>SEARCH_RULE_SET_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the runtime schema of the search rule set.</td>
</tr>
<tr>
<td>SEARCH_RULE_SET_PACKAGE_ID</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the repository package of the search rule set.</td>
</tr>
<tr>
<td>SEARCH_RULE_SET_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the search rule set.</td>
</tr>
<tr>
<td>RULE_NUMBER</td>
<td>INTEGER</td>
<td>Displays the number of the rule that contains the column.</td>
</tr>
<tr>
<td>RULE_ID</td>
<td>NVARCHAR(256)</td>
<td>Displays the unique ID/name of the rule that contains the column.</td>
</tr>
<tr>
<td>SEARCHED_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of a column that is used by a search rule set.</td>
</tr>
<tr>
<td>SEARCHED_COLUMN_CONDITION</td>
<td>NVARCHAR(20)</td>
<td>Displays the condition: EQUALS, NOT EMPTY, or MISSING.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
SEARCHED_COLUMN_VALUE | NVARCHAR(5000) | Displays the value the user input is compared to, if condition is EQUALS.
SEARCHED_COLUMN_VALUE_REPLACED_BY | NVARCHAR(5000) | Displays the user input is replaced by this value if the action is REPLACE.
ACTION | NVARCHAR(20) | Displays the action that is performed when the condition is true: SKIP COLUMN, SKIP RULE, or REPLACE.

### Related Information

SEARCH_RULE_SETS System View [page 1652]
Search, Sort, and Filter Buttons
Search in Table Variables

### 6.1.152 SEQUENCES System View

Provides information about available sequences.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the sequence.</td>
</tr>
<tr>
<td>SEQUENCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the sequence.</td>
</tr>
<tr>
<td>SEQUENCE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the sequence.</td>
</tr>
<tr>
<td>START_NUMBER</td>
<td>DECIMAL</td>
<td>Displays the start number.</td>
</tr>
<tr>
<td>MIN_VALUE</td>
<td>DECIMAL</td>
<td>Displays the minimum value of the sequence in bytes.</td>
</tr>
<tr>
<td>MAX_VALUE</td>
<td>DECIMAL</td>
<td>Displays the maximum value of the sequence in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INCREMENT_BY</td>
<td>DECIMAL</td>
<td>Displays the incremental value.</td>
</tr>
<tr>
<td>IS_CYCLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the sequence starts with the MIN_VALUE after having reached the MAX_VALUE in cases where INCREMENT_BY is greater than 0, or if the sequence starts with the MAX_VALUE after having reached the MIN_VALUE in cases where INCREMENT_BY is less than 0: TRUE/FALSE.</td>
</tr>
<tr>
<td>RESET_BY_QUERY</td>
<td>NCLOB</td>
<td>Displays the reset by query string for the sequence.</td>
</tr>
<tr>
<td>CACHE_SIZE</td>
<td>INTEGER</td>
<td>Displays the cache size of the sequence in bytes.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time of the sequence.</td>
</tr>
</tbody>
</table>

**Related Information**

- **M_SEQUENCES System View [page 2105]**
- **CREATE SEQUENCE Statement (Data Definition) [page 810]**
- **ALTER SEQUENCE Statement (Data Definition) [page 479]**
- **Restart Sequence**

### 6.1.153 SERIES_KEY_COLUMNS System View

Provides information about series key columns for series tables.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the series table name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the series key column name.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the series key column.</td>
</tr>
</tbody>
</table>

**Related Information**

SERIES_TABLES System View [page 1656]
Series Data Functions [page 422]

### 6.1.154 SERIES_TABLES System View

Provides series definitions for tables.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the series table name.</td>
</tr>
<tr>
<td>PERIOD_TYPE_ID</td>
<td>TINYINT</td>
<td>Displays the SQL data type ID of the period columns.</td>
</tr>
<tr>
<td>PERIOD_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the SQL data type name of the period columns.</td>
</tr>
<tr>
<td>IS_EQUIDISTANT</td>
<td>VARCHAR(5)</td>
<td>Displays the series table has equidistant elements: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_INSTANT</td>
<td>VARCHAR(5)</td>
<td>Displays the series represents instants rather than a period with a start/end: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INCREMENT_BY</td>
<td>NVARCHAR(51)</td>
<td>Displays the delta value for an equidistant series or NULL for a non-equidistant series. When the PERIOD_TYPE is a numeric type, a string represents the number. For datetime, a string in an interval form of <code>&lt;number-units&gt;</code>, where units means either year, month, day, hour, minute, or second, represents the number.</td>
</tr>
<tr>
<td>IS_MISSING_ALLOWED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether missing values are allowed in a non-equidistant series: TRUE/FALSE, or NULL for a non-equidistant series.</td>
</tr>
<tr>
<td>PERIOD_START_COLUMN</td>
<td>NVARCHAR(256)</td>
<td>Displays the starting series column name, or NULL if it is an implied or an instant series.</td>
</tr>
<tr>
<td>PERIOD_END_COLUMN</td>
<td>NVARCHAR(256)</td>
<td>Displays the ending series column name, or NULL if it is implied or an instant series.</td>
</tr>
<tr>
<td>PERIOD_INSTANT_COLUMN</td>
<td>NVARCHAR(256)</td>
<td>Displays the instant series column name, or NULL for an interval series.</td>
</tr>
<tr>
<td>MIN_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the minimum value allowed in the series columns.</td>
</tr>
<tr>
<td>MAX_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the maximum value allowed in the series columns.</td>
</tr>
<tr>
<td>CALENDAR_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Reserved.</td>
</tr>
<tr>
<td>CALENDAR_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Reserved.</td>
</tr>
<tr>
<td>CALENDAR_TABLE_OID</td>
<td>BIGINT</td>
<td>Reserved.</td>
</tr>
<tr>
<td>CALENDAR_KEY</td>
<td>NVARCHAR(256)</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

**Related Information**

SERIES_KEY_COLUMNS System View [page 1655]
Series Data Functions [page 422]
### 6.1.155 SESSION_COOKIES System View

Shows information about available session cookies.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user that the session cookie was created for.</td>
</tr>
<tr>
<td>CLIENT_HOST</td>
<td>NVARCHAR(64)</td>
<td>Displays the host name of client machine.</td>
</tr>
<tr>
<td>CLIENT_PID</td>
<td>BIGINT</td>
<td>Displays the client process ID.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the session cookie was created.</td>
</tr>
<tr>
<td>LAST_CONNECT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time that the session cookie was used for successful reconnection.</td>
</tr>
<tr>
<td>SESSION_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the main connection ended.</td>
</tr>
</tbody>
</table>

#### Related Information

- Session Management Statements [page 1205]
- Session Monitoring
- Session-Specific Information for Connections
- Global Session Variables
- M_SESSION_CONTEXT System View [page 2133]
6.1.156 SQLSCRIPT_ANALYZER_RULES System View

Provides information regarding SQLScript Code Analyzer rules.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULE_NAMESPACE</td>
<td>VARCHAR(16)</td>
<td>Displays the namespace for the rule.</td>
</tr>
<tr>
<td>RULE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the rule name.</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>VARCHAR(16)</td>
<td>Displays the classification of the rule: security, consistency, performance, and so on.</td>
</tr>
<tr>
<td>SHORT_DESCRIPTION</td>
<td>NVARCHAR(256)</td>
<td>Displays the short description of the rule.</td>
</tr>
<tr>
<td>LONG_DESCRIPTION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the long description containing the rule background.</td>
</tr>
<tr>
<td>RECOMMENDATION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the recommendation on how to handle the rule.</td>
</tr>
</tbody>
</table>

Related Information

SQLScript Code Analyzer
Limitations in the SQLScript Code Analyzer
6.1.157 SQLSCRIPT_VARIABLE_CACHE System view

Provides information about procedures and functions that have variable caches defined.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema of the target object.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the target object name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the target object type: PROCEDURE/FUNCTION.</td>
</tr>
<tr>
<td>VARIABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the variable to be cached.</td>
</tr>
<tr>
<td>VARIABLE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of variable.</td>
</tr>
<tr>
<td>ENABLE_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the activation mode: ENABLED, DISABLED, or AUTOMATIC.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_SQLSCRIPT_VARIABLE_CACHE System view [page 2142]
- Procedure Result Cache
- Deterministic Procedure Cache
### 6.1.158 STATEMENT_HINTS System View

Provides information about statement hints, including when they were last enabled and/or disabled and by whom.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the MD5 hash value for the STATEMENT_STRING column.</td>
</tr>
<tr>
<td>HINT_STRING</td>
<td>NVARCHAR(5000)</td>
<td>Displays the hint string.</td>
</tr>
<tr>
<td>SYSTEM_HINT_STRING</td>
<td>NVARCHAR(5000)</td>
<td>Displays the system hint string that conflicts with the existing user-defined hint string (if a statement has both a user-defined hint and a system hint associated with it, then only the user-defined hint is effective).</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Specifies comments configured for a statement hint (e.g., ALTER SYSTEM ADD STATEMENT HINT.. COMMENT &lt;comment&gt;..)</td>
</tr>
<tr>
<td>IS_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the hint is enabled. Possible values: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_OVERRIDE</td>
<td>NVARCHAR(5)</td>
<td>Specifies TRUE if an override is specified for the target query (e.g., ALTER SYSTEM ADD STATEMENT HINT.. OVERRIDE...).</td>
</tr>
<tr>
<td>LAST_ENABLE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp for when the hint was last enabled.</td>
</tr>
<tr>
<td>LAST_ENABLE_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the user who enabled the hint.</td>
</tr>
<tr>
<td>LAST_DISABLE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp for when the hint was last disabled.</td>
</tr>
<tr>
<td>LAST_DISABLE_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the user who disabled the hint.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(9)</td>
<td>Displays the type of the target object the hint is being applied to.</td>
</tr>
</tbody>
</table>
Related Information

HINTS System View [page 1585]
HINT Details [page 1134]
ALTER SYSTEM (ENABLE | DISABLE) STATEMENT HINT Statement (System Management) [page 535]
ALTER SYSTEM (ADD | ALTER | REMOVE) STATEMENT HINT Statement (System Management) [page 493]
Statement Hints

6.1.159 STRUCTURED_PRIVILEGES System View

Provides information about available structured (analytic) privileges.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRUCTURED_PRIVILEGE_ID</td>
<td>BIGINT</td>
<td>Displays the object ID of the structured privilege.</td>
</tr>
<tr>
<td>STRUCTURED_PRIVILEGE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the privilege.</td>
</tr>
<tr>
<td>STRUCTURED_PRIVILEGE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the structured privilege.</td>
</tr>
<tr>
<td>RESTRICTION_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the type of restriction: CUBERESTRICTION, ACTIVITYRESTRICTION, VALIDITYRESTRICTION, or DIMENSIONRESTRICTION.</td>
</tr>
<tr>
<td>DIMENSION_ATTRIBUTE</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the dimension attribute.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FILTER_ID</td>
<td>BIGINT</td>
<td>Displays the number of filters needed to combine all operators/operands belonging to one filter.</td>
</tr>
<tr>
<td>FILTER_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of filter: STATIC/DYNAMIC.</td>
</tr>
<tr>
<td>NEGATED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the operator is negated in the filter: TRUE/FALSE.</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>VARCHAR(16)</td>
<td>Displays the type of operator: CONTAINS PATTERN, BETWEEN, EQUAL, IN, LESS THAN, LESS EQUAL, GREATER THAN, or GREATER EQUAL.</td>
</tr>
<tr>
<td>OPERAND_ORDER</td>
<td>INTEGER</td>
<td>Displays the sequence of the operands per filter ID.</td>
</tr>
<tr>
<td>OPERAND</td>
<td>NCLOB</td>
<td>Displays the value that the operator is compared to.</td>
</tr>
</tbody>
</table>

**Related Information**

- CREATE STRUCTURED PRIVILEGE Statement (Access Control) [page 822]
- ALTER STRUCTURED PRIVILEGE Statement (Access Control) [page 491]
- DROP STRUCTURED PRIVILEGE Statement (Access Control) [page 978]
- PRIVILEGES System View [page 1611]
- EFFECTIVE_STRUCTURED_PRIVILEGES System View [page 1555]
- Analytic Privileges
  Privileges
  Managing Privileges
### ST_GEOMETRY_COLUMNS System View

Provides information about spatial columns.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema to which the table containing the spatial column belongs.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table containing the spatial column.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the spatial column.</td>
</tr>
<tr>
<td>SRS_ID</td>
<td>INTEGER</td>
<td>Displays the numeric identifier (SRID) for the spatial reference system (SRS) associated with the spatial column.</td>
</tr>
<tr>
<td>SRS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the SRS associated with the spatial column.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the spatial data type.</td>
</tr>
<tr>
<td>COORD_DIMENSION</td>
<td>INTEGER</td>
<td>Displays the dimension of the spatial column.</td>
</tr>
<tr>
<td>INTERNAL_LAYOUT</td>
<td>VARCHAR(13)</td>
<td>Displays the internal geometry layout: PLAIN or HILBERT CURVE. For details see the column definition and column configuration.</td>
</tr>
<tr>
<td>SPATIAL_INDEX_PREFERENCE</td>
<td>VARCHAR(8)</td>
<td>Displays the spatial index preference: DEFAULT, RDICT, or RTREE. For details see the column definition and the column configuration.</td>
</tr>
<tr>
<td>VALIDATION</td>
<td>VARCHAR(4)</td>
<td>Displays the geometry validation level: NONE (checked for bad representation)/FULL (checked for all known invalid representations as defined by the OGC standard).</td>
</tr>
</tbody>
</table>
### Additional Information

Spatial columns in virtual tables are not shown in the system view ST_GEOMETRY_COLUMNS.

### Related Information

- **ST_SPATIAL_REFERENCE(System View) View [page 1665]**
- **ST_UNITS_OF_MEASURE System View [page 1668]**
- **Spatial Data Types [page 49]**
- **TABLE_COLUMNS System View [page 1675]**
- **VIEW_COLUMNS System View [page 1705]**
- **Support for SAP HANA Spatial**

### 6.1.161 ST_SPATIAL_REFERENCE_SYSTEMS System View

Provides information about spatial reference systems.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the owner name of the spatial reference system (SRS).</td>
</tr>
<tr>
<td>SRS_ID</td>
<td>INTEGER</td>
<td>Displays the numeric identifier (SRID) of this spatial reference system.</td>
</tr>
<tr>
<td>SRS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the spatial reference system.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ROUND_EARTH</td>
<td>VARCHAR(7)</td>
<td>Indicates whether the SRS type is ROUND EARTH (TRUE) or PLANAR (FALSE).</td>
</tr>
<tr>
<td>AXIS_ORDER</td>
<td>VARCHAR(12)</td>
<td>Displays how the database server interprets points with regards to latitude and longitude: x/y/z/m, long/lat/z/m, lat/long/z/m.</td>
</tr>
<tr>
<td>SNAP_TO_GRID</td>
<td>DOUBLE</td>
<td>Defines the granularity of the grid, in the LINEAR_UNIT_OF_MEASURE, that geometries are snapped to on creation.</td>
</tr>
<tr>
<td>TOLERANCE</td>
<td>DOUBLE</td>
<td>Defines the precision to use when comparing points.</td>
</tr>
<tr>
<td>SEMI_MAJOR_AXIS</td>
<td>DOUBLE</td>
<td>Displays the distance from the center of the ellipsoid to the equator for a ROUND EARTH SRS.</td>
</tr>
<tr>
<td>SEMI_MINOR_AXIS</td>
<td>DOUBLE</td>
<td>Displays the distance from the center of the ellipsoid to the poles for a ROUND EARTH SRS.</td>
</tr>
<tr>
<td>INV_FLATTENING</td>
<td>DOUBLE</td>
<td>Displays the inverse flattening used for the ellipsoid in a ROUND EARTH SRS.</td>
</tr>
<tr>
<td>MIN_X</td>
<td>DOUBLE</td>
<td>Displays the minimum x value allowed in coordinates.</td>
</tr>
<tr>
<td>MAX_X</td>
<td>DOUBLE</td>
<td>Displays the maximum x value allowed in coordinates.</td>
</tr>
<tr>
<td>MIN_Y</td>
<td>DOUBLE</td>
<td>Displays the minimum y value allowed in coordinates.</td>
</tr>
<tr>
<td>MAX_Y</td>
<td>DOUBLE</td>
<td>Displays the maximum y value allowed in coordinates.</td>
</tr>
<tr>
<td>MIN_Z</td>
<td>DOUBLE</td>
<td>Displays the minimum z value allowed in coordinates.</td>
</tr>
<tr>
<td>MAX_Z</td>
<td>DOUBLE</td>
<td>Displays the maximum z value allowed in coordinates.</td>
</tr>
<tr>
<td>ORGANIZATION</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the organization that created the coordinate system used by this spatial reference system.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ORGANIZATION_COORDSYS_ID</td>
<td>INTEGER</td>
<td>Displays the ID given to the coordinate system by the organization that created it.</td>
</tr>
<tr>
<td>SRS_TYPE</td>
<td>VARCHAR(11)</td>
<td>Displays the type of SRS as defined by the SQL/MM standard: GEOGRAPHIC, PROJECTED, or ENGINEERING.</td>
</tr>
<tr>
<td>LINEAR_UNIT_OF_MEASURE</td>
<td>NVARCHAR(256)</td>
<td>Displays the linear unit of measure used by this spatial reference system.</td>
</tr>
<tr>
<td>ANGULAR_UNIT_OF_MEASURE</td>
<td>NVARCHAR(256)</td>
<td>Displays the angular unit of measure used by this spatial reference system.</td>
</tr>
<tr>
<td>POLYGON_FORMAT</td>
<td>VARCHAR(16)</td>
<td>Displays the orientation of the rings in a polygon: COUNTERCLOCKWISE, CLOCKWISE, or EVENODD.</td>
</tr>
<tr>
<td>STORAGE_FORMAT</td>
<td>VARCHAR(8)</td>
<td>Indicates whether the data is stored in normalized format (INTERNAL), un-normalized format (ORIGINAL), or both (MIXED).</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NVARCHAR(256)</td>
<td>Displays the WKT definition of the spatial reference system in the format defined by the OGC standard.</td>
</tr>
<tr>
<td>TRANSFORM_DEFINITION</td>
<td>NVARCHAR(256)</td>
<td>Displays the transform definition settings for use when transforming data from this SRS to another.</td>
</tr>
</tbody>
</table>

**Related Information**

- **ST_GEOMETRY_COLUMNS** System View [page 1664]
- **ST_UNITS_OF_MEASURE** System View [page 1668]
- Spatial Data Types [page 49]
- Support for SAP HANA Spatial
- HANA Spatial Support
6.1.162  ST_UNITS_OF_MEASURE System View

Provides information about spatial units of measure.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the owner name of this unit of measure.</td>
</tr>
<tr>
<td>UNIT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of this unit of measure.</td>
</tr>
<tr>
<td>UNIT_TYPE</td>
<td>VARCHAR(7)</td>
<td>Displays the type of this unit of measure: ANGULAR/LINEAR.</td>
</tr>
<tr>
<td>CONVERSION_FACTOR</td>
<td>DOUBLE</td>
<td>Displays the conversion factor of this unit of measure.</td>
</tr>
</tbody>
</table>

Related Information

Spatial Data Types [page 49]
ST_GEOMETRY_COLUMNS System View [page 1664]
ST_SPATIAL_REFERENCE_SYSTEMS System View [page 1665]

6.1.163  SYNONYMS System View

Lists available synonyms.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the synonym.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>SYNONYM_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the synonym.</td>
</tr>
<tr>
<td>SYNONYM_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the synonym.</td>
</tr>
<tr>
<td>OBJECT_DATABASE</td>
<td>NVARCHAR(256)</td>
<td>Displays the database name of the referenced object.</td>
</tr>
<tr>
<td>OBJECT_SCHEMA</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the referenced object.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the referenced object.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of the referenced object.</td>
</tr>
<tr>
<td>IS_COLUMN_OBJECT</td>
<td>VARCHAR(5)</td>
<td>Displays whether this view is a column object or not: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the synonym is valid or not: TRUE/FALSE. This value becomes FALSE when its base object is dropped.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time of the synonym.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE SYNONYM Statement (Data Definition) [page 823]
DROP SYNONYM Statement (Data Definition) [page 979]
CREATE SCHEMA SYNONYM Statement (Data Definition) [page 808]
DROP SCHEMA SYNONYM Statement (Data Definition) [page 972]

**6.1.164 TABLES System View**

Provides information about tables in the database.
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the table description.</td>
</tr>
<tr>
<td>FIXED_PART_SIZE</td>
<td>SMALLINT</td>
<td>Displays the fixed part size of the table.</td>
</tr>
<tr>
<td>IS_LOGGED</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not logging was on for the table at last restart time: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_SYSTEM_TABLE</td>
<td>VARCHAR(5)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>IS_COLUMN_TABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is a column table: TRUE/FALSE.</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of the table: ROW, COLUMN, VIRTUAL, EXTENDED, or COLLECTION.</td>
</tr>
<tr>
<td>IS_INSERT_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is an insert-only table: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_TENANT_SHARED_DATA</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table can be shared among other instances: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_TENANT_SHARED_METADATA</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is a global table: TRUE/FALSE.</td>
</tr>
<tr>
<td>SESSION_TYPE</td>
<td>VARCHAR(7)</td>
<td>Displays the session type: NONE, SIMPLE, or HISTORY. For HISTORY, time travel is possible.</td>
</tr>
<tr>
<td>IS_TEMPORARY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is a temporary table: TRUE/FALSE.</td>
</tr>
<tr>
<td>TEMPORARY_TABLE_TYPE</td>
<td>VARCHAR(10)</td>
<td>Displays the temporary table type.</td>
</tr>
<tr>
<td>COMMIT_ACTION</td>
<td>VARCHAR(8)</td>
<td>Displays the setting for on commit deletion for global temporary tables.</td>
</tr>
<tr>
<td>IS_USER_DEFINED_TYPE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not this is a user-defined table type: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HAS_PRIMARY_KEY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table has a primary key: TRUE/FALSE. Only valid for column store tables.</td>
</tr>
<tr>
<td>PARTITION_SPEC</td>
<td>NCLOB</td>
<td>Displays the detail specification of table partitioning. Only valid for column store tables.</td>
</tr>
<tr>
<td>USES_EXTKEY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table uses an external key: TRUE/FALSE. Only valid for column store tables.</td>
</tr>
<tr>
<td>AUTO_MERGE_ON</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not automatic delta table merge is activated for the table: TRUE/FALSE. Only valid for column store tables.</td>
</tr>
<tr>
<td>USES_DIMFN_CACHE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table uses the DimFunctionCache feature: TRUE/FALSE. Only valid for column store tables.</td>
</tr>
<tr>
<td>IS_PUBLIC</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table is public: TRUE/FALSE. Only valid for column store tables.</td>
</tr>
<tr>
<td>AUTO_OPTIMIZE_COMPRESSION_ON</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not automatic optimize compression is activated for the table: TRUE/FALSE. Only valid for column store tables.</td>
</tr>
<tr>
<td>COMPRESSED_EXTKEY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table uses a compressed external key: TRUE/FALSE. Only valid for column store tables.</td>
</tr>
<tr>
<td>HAS_TEXT_FIELDS</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table has at least one column of type TEXT: TRUE/FALSE. Only valid for column store tables.</td>
</tr>
<tr>
<td>USES_QUEUE_TABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table uses a queue table: TRUE/FALSE. Only valid for column store tables.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_PRELOAD</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table uses preloading: TRUE/FALSE. Only valid for column store tables. Preload information can be NULL for row store, virtual columns, or tables on extended storage.</td>
</tr>
<tr>
<td>IS_PARTIAL_PRELOAD</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table uses partial preloading: TRUE/FALSE. Only valid for column store tables. Preload information can be NULL for row store, virtual columns, or tables on extended storage.</td>
</tr>
<tr>
<td>UNLOAD_PRIORITY</td>
<td>TINYINT</td>
<td>Displays the unload priority of the table: 0 - 9 means not unloadable, 1 means latest unload, and 9 means earliest unload.</td>
</tr>
<tr>
<td>HAS_SCHEMA_FLEXIBILITY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table has schema flexibility: TRUE/FALSE. Only valid for column store tables.</td>
</tr>
<tr>
<td>IS_REPLICA</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table is a replica: TRUE/FALSE.</td>
</tr>
<tr>
<td>HAS_STRUCTURED_PRIVILEGE_CHECK</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table is registered for a structured privilege check: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_SERIES_TABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table is a series table: TRUE/FALSE.</td>
</tr>
<tr>
<td>ROW_ORDER_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the row order type: NULL/BY VALUE.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time of the table.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TEMPORAL_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the type of temporal table:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TEMPORAL - a table with system versioning enabled. Tables of this type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>have a corresponding entry in the TEMPORAL_TABLES system table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• HISTORY - the history table associated with a system-versioned table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• NULL - all other types of tables.</td>
</tr>
<tr>
<td>HAS_MASKED_COLUMNS</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the table has a mask definition for at least one</td>
</tr>
<tr>
<td></td>
<td></td>
<td>column: TRUE/FALSE.</td>
</tr>
<tr>
<td>MASK_MODE</td>
<td>NVARCHAR(12)</td>
<td>Displays the mask mode to be applied if the table has masked columns:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFAULT or SESSION USER; otherwise NULL.</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY</td>
<td>VARCHAR(5)</td>
<td>Displays the user-specified persistent memory preference: TRUE (persistent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>memory is ON)/FALSE (persistent memory OFF). If the user has not specified a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>persistent memory preference, or if the preference is set to the default,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>then the value is NULL.</td>
</tr>
<tr>
<td>HAS_RECORD_COMMIT_TIMESTAMP</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is tracking commit timestamps: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_REPLICATION_LOG_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table’s replication log is collected. Values are TRUE/</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE.</td>
</tr>
</tbody>
</table>
| NUMA_NODE_INDEXES         | VARCHAR (1024)| Displays a comma-separated list of user-specified logical NUMA node indexes for
<p>|                           |               | the table. For each column entry in this view there are preferences (if set); |
|                           |               | otherwise, the value is NULL.                                               |
|                           |               | The elements of the list are either individual nodes indexes or ranges of     |
|                           |               | index nodes, or a mixture of both. For example, the value 0, 1 TO 3, 6      |
|                           |               | indicates node indexes 0, 1, 2, 3, and 6.                                   |</p>
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS_MOVABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table can be moved to another location: TRUE/ FALSE.</td>
</tr>
<tr>
<td>LOAD_UNIT</td>
<td>VARCHAR(7)</td>
<td>Displays the load unit for the table partition: TABLE, PAGE (when the partition uses paged attributes), COLUMN, or DEFAULT.</td>
</tr>
</tbody>
</table>

**Related Information**

TABLE_COLUMNS System View [page 1675]
TABLE_COLUMNS_ODBC System View [page 1679]
TABLE_GROUPS System View [page 1680]
TABLE_PARTITIONS System View [page 1681]
TABLE_PLACEMENT System View [page 1684]
TABLE_REPLICAS System View [page 1686]
TEMPORAL_TABLES System View [page 1690]
M_TABLES System View [page 2200]
M_TABLE_LOCATIONS System View [page 2204]
M_TABLE_LOCKS System View [page 2206]
M_TABLE_SNAPSHOTS System View [page 2226]
M_TABLE_STATISTICS System View [page 2227]
M_TABLE_STATISTICS_RESET System View [page 2228]
M_TABLE_VIRTUAL_FILES System View [page 2229]
CREATE TABLE Statement (Data Definition) [page 825]
ALTER TABLE Statement (Data Definition) [page 589]
LOCK TABLE Statement (Transaction Management) [page 1056]
RENAME TABLE Statement (Data Definition) [page 1094]
TRUNCATE TABLE Statement (Data Manipulation) [page 1178]
DROP TABLE Statement (Data Definition) [page 980]
Managing Tables
Multistore Tables
Emptiness Check for Tables and Table Variables
Table Parameter
Save Table
Table Variables
Table Types
### 6.1.165 TABLE_COLUMNS System View

Provides information about available table columns.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Indicates the ordinal position of the column in a record.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the SQL data type ID of the column.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the SQL data type name of the column.</td>
</tr>
<tr>
<td>OFFSET</td>
<td>SMALLINT</td>
<td>Displays the offset of the column in the record.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Indicates the number of chars for CHAR types, number of max digits for numeric types, number of chars for date-time types, and number of bytes for LOB types.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Indicates the numeric types: the maximum number of digits to the right of the decimal point; time, timestamp: the decimal digits are defined as the number of digits to the right of the decimal point in the second’s component of the data.</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is allowed to accept null values: TRUE or FALSE.</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the default value of the column.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COLLATION</td>
<td>NVARCHAR(256)</td>
<td>Displays the collation used for this column.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the description for this column.</td>
</tr>
<tr>
<td>MAX_VALUE</td>
<td>VARCHAR(1)</td>
<td>Deprecated: no longer set.</td>
</tr>
<tr>
<td>MIN_VALUE</td>
<td>VARCHAR(1)</td>
<td>Deprecated: no longer set.</td>
</tr>
<tr>
<td>CS_DATA_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the column store data type ID.</td>
</tr>
<tr>
<td>CS_DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the column store data type name.</td>
</tr>
<tr>
<td>DDIC_DATA_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the DDIC data type ID.</td>
</tr>
<tr>
<td>DDIC_DATA_TYPE_NAME</td>
<td>VARCHAR(7)</td>
<td>Displays the DDIC data type name.</td>
</tr>
<tr>
<td>COMPRESSION_TYPE</td>
<td>VARCHAR(9)</td>
<td>Indicates the type of compression: DEFAULT, PREFIXED, SPARSE, CLUSTERED, SPARSE, CLUSTERED, INDIRECT, RLE, and LINEAR RLE. The default value for the column is shown. The actual value can be checked in the system view M_CS_COLUMNS. For row store columns, NONE is shown.</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>VARCHAR(4)</td>
<td>Indicates the type of index: NONE or FULL.</td>
</tr>
<tr>
<td>COLUMN_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the column.</td>
</tr>
<tr>
<td>PRELOAD</td>
<td>VARCHAR(5)</td>
<td>Displays if the column is preloaded: TRUE or FALSE. Preload information can be NULL in case of row store, virtual columns, or tables on extended storage.</td>
</tr>
<tr>
<td>GENERATED_ALWAYS_AS</td>
<td>NVARCHAR(1000)</td>
<td>Displays the expression of the column created by GENERATED... AS.</td>
</tr>
<tr>
<td>HAS_SCHEMA_FLEXIBILITY</td>
<td>VARCHAR(5)</td>
<td>Displays if the column has schema flexibility: TRUE or FALSE.</td>
</tr>
<tr>
<td>FUZZY_SEARCH_INDEX</td>
<td>VARCHAR(5)</td>
<td>Displays if the column has a fuzzy search index: TRUE or FALSE.</td>
</tr>
<tr>
<td>FUZZY_SEARCH_MODE</td>
<td>VARCHAR(32)</td>
<td>Displays the fuzzy search mode.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MEMORY_THRESHOLD</td>
<td>INTEGER</td>
<td>Displays the memory threshold in bytes for LOB types.</td>
</tr>
<tr>
<td>LOAD_UNIT</td>
<td>VARCHAR(7)</td>
<td>Displays the unit of column data loading: TABLE, COLUMN, PAGE, or DEFAULT.</td>
</tr>
<tr>
<td>GENERATION_TYPE</td>
<td>VARCHAR(22)</td>
<td>Displays ALWAYS AS if the column is generation column, ALWAYS AS IDENTITY, BY DEFAULT AS IDENTITY if the column is an identity column whose values are always generated or generated by default, and NULL otherwise. For system-versioning columns, the value 'ALWAYS AS ROW START' indicates a column that was defined as GENERATED ALWAYS AS ROW START, while the value 'ALWAYS AS ROW END' indicates a column that was defined as GENERATED ALWAYS AS ROW END.</td>
</tr>
<tr>
<td>IS_CACHABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is part of cached data: TRUE or FALSE.</td>
</tr>
<tr>
<td>IS_CACHE_KEY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is part of cache key: TRUE or FALSE.</td>
</tr>
<tr>
<td>ROW_ORDER_POSITION</td>
<td>SMALLINT</td>
<td>Displays the row order position.</td>
</tr>
<tr>
<td>IS_HIDDEN</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is hidden: TRUE or FALSE.</td>
</tr>
<tr>
<td>IS_MASKED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is masked: TRUE or FALSE.</td>
</tr>
<tr>
<td>MASK_EXPRESSION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the mask expression. It is only visible to users with the DATA ADMIN and CATALOG READ privileges, the SELECT METADATA privilege on the schema, and who are the owner of the table or schema.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CLIENTSIDE_ENCRYPTION_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays whether the column is undergoing encryption (ENCRYPTING), decryption (DECRYPTING), re-encryption with a different column encryption key (REENCRYPTING), re-encryption with the latest version of the existing column encryption key (ACTIVATING), is already encrypted (ENCRYPTED), or is in clear-text (NULL).</td>
</tr>
<tr>
<td>CLIENTSIDE_ENCRYPTION_COLUMN_KEY_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the UUID of the column encryption key used to encrypt the column. The value is NULL if the column is not encrypted.</td>
</tr>
<tr>
<td>CLIENTSIDE_ENCRYPTION_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the encryption mode. Possible values are RANDOM, DETERMINISTIC, or NULL (not encrypted).</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY</td>
<td>VARCHAR(5)</td>
<td>Displays the user-specified persistent memory preference: TRUE (persistent memory is ON), or FALSE (persistent memory OFF). If the user has not specified a persistent memory preference, or if the preference is set to the default, then the value is NULL.</td>
</tr>
</tbody>
</table>
| NUMA_NODE_INDEXES                               | VARCHAR(1024)| Displays a comma-separated list of user-specified logical NUMA node indexes for the table columns. For each column entry in this view there will be preferences (if set); otherwise, the value is NULL.  

The elements of the list are either individual nodes indexes or ranges of index nodes, or a mixture of both. For example, the value 0, 1 TO 3, 6 indicates node indexes 0, 1, 2, 3, and 6. |

**Related Information**

- TABLE_COLUMNS_ODBC System View [page 1679]
- TABLES System View [page 1669]
- RENAME COLUMN Statement (Data Definition) [page 1088]
- COLUMNS System View - Deprecated [page 1520]
6.1.166  TABLE_COLUMNS_ODBC System View

Provides information about available table columns.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the SQL data type ID of the column.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the SQL data type name of the column.</td>
</tr>
<tr>
<td>COLUMN_SIZE</td>
<td>INTEGER</td>
<td>Displays the total number of characters required to display the value when it is converted to characters.</td>
</tr>
<tr>
<td>BUFFER_LENGTH</td>
<td>INTEGER</td>
<td>Displays the length, in bytes, required to transfer the value.</td>
</tr>
<tr>
<td>DECIMAL_DIGITS</td>
<td>INTEGER</td>
<td>Displays the total number of significant digits to the right of the decimal point.</td>
</tr>
<tr>
<td>NUM_PREC_RADIX</td>
<td>INTEGER</td>
<td>Displays how to interpret columns COLUMN_SIZE and DECIMAL_DIGITS. For numeric data types either 10 or 2.</td>
</tr>
<tr>
<td>NULLABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is allowed to accept null value: TRUE/FALSE.</td>
</tr>
<tr>
<td>COLUMN_DEF</td>
<td>NVARCHAR(5000)</td>
<td>Displays the default value of the column.</td>
</tr>
</tbody>
</table>
### Related Information

- TABLES System View [page 1669]
- TABLE_COLUMNS System View [page 1675]
- COLUMNS System View - Deprecated [page 1520]

### 6.1.167 TABLE_GROUPS System View

Provides an overview of table group relationships.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>GROUP_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the group type.</td>
</tr>
<tr>
<td>SUBTYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the subtype.</td>
</tr>
<tr>
<td>GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the group name.</td>
</tr>
</tbody>
</table>
### Related Information

**TABLES System View** [page 1669]
Table Classification (Groups)
Group Advisor
Table Placement

### 6.1.168 TABLE_PARTITIONS System View

Partition-specific information for partitioned tables.

#### Structure

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>NODE_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the partition node.</td>
</tr>
<tr>
<td>PARENT_NODE_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the parent node.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Returns the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For replicated tables, the part ID is 1 for the original table and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>currently in progress.</td>
</tr>
</tbody>
</table>

| IS_GROUP_LEAD   | VARCHAR(5)    | Determines the leading table within a group.                                 |

Determined the leading table within a group.
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL_1_PARTITION</td>
<td>INTEGER</td>
<td>Displays the first level partition ID. Possible values are 1 through the number of first-level partitions. First-level partitions (those used in ALTER TABLE...MOVE) may have subpartitions. Displays the logical partition ID. Possible values are 1 through the number of partitions for partitioned tables. This is the ID shown in all monitoring views.</td>
</tr>
<tr>
<td>LEVEL_2_PARTITION</td>
<td>INTEGER</td>
<td>Displays the subpartition ID. Possible values are 0 for tables without multilevel partitioning and 1 through the number of subpartitions for partitioned tables with multilevel partitioning.</td>
</tr>
<tr>
<td>LEVEL_1_RANGE_MIN_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the minimum value of the range partition at the first level for a range-partitioned table; empty otherwise.</td>
</tr>
<tr>
<td>LEVEL_1_RANGE_MAX_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the exclusive maximum value of the range partition at the first level for a range-partitioned table; empty otherwise. This value is not included in the range.</td>
</tr>
<tr>
<td>LEVEL_2_RANGE_MIN_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the minimum value of the range partition at the second level for a range-partitioned table; empty otherwise.</td>
</tr>
<tr>
<td>LEVEL_2_RANGE_MAX_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the exclusive maximum value of the range partition at the second level for a range-partitioned table; empty otherwise. This value is not included in the range.</td>
</tr>
<tr>
<td>IS_CURRENT</td>
<td>VARCHAR(5)</td>
<td>Displays whether the partition is the current partition: TRUE/FALSE.</td>
</tr>
<tr>
<td>UNIQUE_CONSTRAINTS</td>
<td>VARCHAR(5)</td>
<td>Displays if unique constraints are checked for this partition or not: TRUE/FALSE.</td>
</tr>
<tr>
<td>LOAD_UNIT</td>
<td>VARCHAR(7)</td>
<td>Displays the load unit for partition: PAGE (when the partition uses page loadable storage), COLUMN (when the partition uses column loadable storage), or DEFAULT.</td>
</tr>
<tr>
<td>INSERT</td>
<td>VARCHAR(5)</td>
<td>Displays whether INSERT statements are allowed on the partition: TRUE/FALSE.</td>
</tr>
<tr>
<td>STORAGE_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the storage type for the partition: DEFAULT/EXTENDED.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY</td>
<td>VARCHAR(5)</td>
<td>Displays the user-specified persistent memory preference: TRUE (persistent memory is ON)/FALSE (persistent memory OFF). If the user has not specified a persistent memory preference, or if the preference is set to the default, then the value is NULL.</td>
</tr>
<tr>
<td>NUMA_NODE_INDEXES</td>
<td>VARCHAR(1024)</td>
<td>Displays a comma-separated list of user-specified logical NUMA node indexes for the table partition. The elements of the list are either individual nodes indexes or ranges of index nodes, or a mixture of both. For example, the value 0, 1 TO 3, 6 indicates node indexes 0, 1, 2, 3, and 6.</td>
</tr>
<tr>
<td>GROUP_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the group type.</td>
</tr>
<tr>
<td>SUBTYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the group subtype.</td>
</tr>
<tr>
<td>GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the group name.</td>
</tr>
</tbody>
</table>

**Additional Information**

You must have the DATA ADMIN system privilege to see table information, unless you own or have SELECT permission on the tables.

**Related Information**

- TABLES System View [page 1669]
- M_TABLE_PARTITIONS System View [page 2209]
- M_TABLE_PARTITION_STATISTICS System View [page 2207]
- Heterogeneous Create Partition Clauses [page 869]
- Heterogeneous Alter Partition Clauses [page 640]
- Non-heterogeneous Create Partition Clauses [page 863]
- Non-heterogeneous Alter Partition Clauses [page 630]
- Table Partitioning
- Partition a Non-Partitioned Table
- Change a Partitioned Table into a Non-Partitioned Table
- Optimize Table Partitioning
- Move Table Partitions to Another Host
6.1.169 TABLE_PLACEMENT System View

Provides table placement information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the group name.</td>
</tr>
<tr>
<td>GROUP_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the group type.</td>
</tr>
<tr>
<td>SUBTYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the subtype.</td>
</tr>
<tr>
<td>MIN_ROWS_FOR_PARTITIONING</td>
<td>BIGINT</td>
<td>Displays the minimum number of rows for partitioning.</td>
</tr>
<tr>
<td>INITIAL_PARTITIONS</td>
<td>INTEGER</td>
<td>Displays the initial number of partitions.</td>
</tr>
<tr>
<td>REPARTITIONING_THRESHOLD</td>
<td>BIGINT</td>
<td>Displays the repartitioning threshold.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>VARCHAR(256)</td>
<td>Displays the location.</td>
</tr>
<tr>
<td>DYNAMIC_RANGE_THRESHOLD</td>
<td>BIGINT</td>
<td>Displays the dynamic range threshold.</td>
</tr>
<tr>
<td>SAME_PARTITION_COUNT</td>
<td>VARCHAR(5)</td>
<td>Displays whether to use the same count of partitions for each table belonging to the group: TRUE/FALSE.</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the data is loaded in persistent memory: TRUE/FALSE.</td>
</tr>
<tr>
<td>PAGE_LOADABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the data is page loadable: TRUE/FALSE.</td>
</tr>
<tr>
<td>REPLICA_COUNT</td>
<td>INTEGER</td>
<td>Sets the required number of replicas.</td>
</tr>
<tr>
<td>NUMA_NODE_INDEXES</td>
<td>NVARCHAR(5000)</td>
<td>Sets the allowed numa nodes.</td>
</tr>
</tbody>
</table>

Related Information

ALTER SYSTEM ALTER TABLE PLACEMENT Statement (System Management) [page 508]
6.1.170 TABLE_PLACEMENT_LOCATIONS System View

Provides table placement location information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATION_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the location name.</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>VARCHAR(256)</td>
<td>Displays the list of volume IDs or synonyms which are added to the respective location name.</td>
</tr>
<tr>
<td>EXCLUDE</td>
<td>VARCHAR(256)</td>
<td>Displays the list of volume IDs or synonyms which are removed from the respective location name.</td>
</tr>
</tbody>
</table>

Related Information

ALTER SYSTEM ALTER TABLE PLACEMENT Statement (System Management) [page 508]
Table Placement
Table Placement Rules
Add or Edit a Table Placement Rule
Copy a Table Placement Rule
Manage Table Placement Rule Locations
TABLES System View [page 1669]
TABLE_PLACEMENT System View [page 1684]
TABLE_PLACEMENT_LOCATIONS System View [page 1685]
M_TABLE_PLACEMENT_LOCATIONS System View [page 2217]
M_TABLE_LOCATIONS System View [page 2204]
6.1.171 TABLE_REPLICAS System View

Provides information about replicated tables and their replicas.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>SOURCE_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the source table name.</td>
</tr>
<tr>
<td>SOURCE_TABLE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the source table type.</td>
</tr>
<tr>
<td>REPLICA_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the replica schema name.</td>
</tr>
<tr>
<td>REPLICA_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the replica name.</td>
</tr>
<tr>
<td>REPLICA_TABLE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the replica table type.</td>
</tr>
<tr>
<td>REPLICA_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the replica type.</td>
</tr>
<tr>
<td>HAS_DIFFERENT_PARTITIONS</td>
<td>VARCHAR(5)</td>
<td>Displays whether the replica table has different partitions from the source table.</td>
</tr>
<tr>
<td>HAS_DIFFERENT_COLUMNS</td>
<td>VARCHAR(5)</td>
<td>Displays whether the replica table has different columns from the source table.</td>
</tr>
</tbody>
</table>

Related Information

ALTER SYSTEM {ENABLE | DISABLE} ALL [ASYNCHRONOUS | SYNCHRONOUS] TABLE REPLICAS Statement (System Management) [page 534]
M_TABLE_REPLICAS System View [page 2220]
M_TABLE_REPLICAS_RESET System View [page 2222]
Table Replication
Operations for Asynchronous Table Replication
Configure Asynchronous Table Replication
Table Replication Limitations
Asynchronous Table Replication
6.1.172 TASKS System View

Provides information regarding table tasks.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASK_OID</td>
<td>BIGINT</td>
<td>Displays the unique identifier for the task.</td>
</tr>
<tr>
<td>TASK_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the task.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema that the task was created in.</td>
</tr>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the owner of the task.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size of loaded task in bytes.</td>
</tr>
<tr>
<td>TASK_TYPE</td>
<td>NVARCHAR(64)</td>
<td>Displays the type of task derived from the task plan.</td>
</tr>
<tr>
<td>PLAN_VERSION</td>
<td>NVARCHAR(32)</td>
<td>Displays the version of the task plan.</td>
</tr>
<tr>
<td>PLAN</td>
<td>NCLOB</td>
<td>Displays the task plan used to define the task. This value is NULL if the task was created using a procedure.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(256)</td>
<td>Displays the description of the task from the task plan.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HAS_TABLE_TYPE_INPUT</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the task is modeled with a table type as input: TRUE/FALSE. If TRUE, this means that data would need to be pushed at execution time.</td>
</tr>
<tr>
<td>HAS_SDQ</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the task contains SDQ (Smart Data Quality) functionality: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_REALTIME_TASK</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the specified task is a realtime task.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the task is in a valid state: TRUE/FALSE. FALSE indicates that the task has been invalidated by a dependency.</td>
</tr>
<tr>
<td>IS_READ_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the task is read only, meaning it has only table type outputs: TRUE/FALSE. FALSE indicates that the task writes to non-table-type outputs.</td>
</tr>
<tr>
<td>PROCEDURE_SCHEMA</td>
<td>NVARCHAR(256)</td>
<td>If the task was created with a procedure instead of a plan, contains the schema name of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>If the task was created with a procedure instead of a plan, contains the name of the stored procedure.</td>
</tr>
<tr>
<td>INPUT_PARAMETER_COUNT</td>
<td>SMALLINT</td>
<td>Displays the number of input (tableType) parameters.</td>
</tr>
<tr>
<td>OUTPUT_PARAMETER_COUNT</td>
<td>SMALLINT</td>
<td>Displays the number of output (tableType) parameters.</td>
</tr>
<tr>
<td>SQL_SECURITY</td>
<td>VARCHAR(7)</td>
<td>Displays the security model for the task: DEFINER/INVOKER.</td>
</tr>
</tbody>
</table>

**Related Information**

- TASK_PARAMETERS System View [page 1689]
- M_TASKS System View [page 2230]
- HOST_TASKS View (Embedded Statistics Service)
6.1.173 TASK_PARAMETERS System View

Provides task parameter information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema that the task was created in.</td>
</tr>
<tr>
<td>TASK_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the task.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the parameter.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the position of the parameter.</td>
</tr>
<tr>
<td>TABLE_TYPE_SCHEMA</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema that the table type was created in.</td>
</tr>
<tr>
<td>TABLE_TYPE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table type.</td>
</tr>
<tr>
<td>PARAMETER_TYPE</td>
<td>VARCHAR(7)</td>
<td>Displays the parameter type: IN/OUT.</td>
</tr>
</tbody>
</table>

Related Information

TASKS System View [page 1687]  
M_TASKS System View [page 2230]  
HOST_TASKS View (Embedded Statistics Service)  
START TASK Statement [Smart Data Integration]  
CANCEL TASK Statement [Smart Data Integration]  
TASKS System View [Smart Data Integration]  
TASK_EXECUTIONS System View [Smart Data Integration]  
HOST_TASKS View (Embedded Statistics Service)
6.1.174 TEMPORAL_TABLES System View

Provides information about system-versioned tables, history tables, and application-time period tables.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>PERIOD_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the period name.</td>
</tr>
<tr>
<td>PERIOD_START_COLUMN</td>
<td>NVARCHAR(256)</td>
<td>Displays the period start column name.</td>
</tr>
<tr>
<td>PERIOD_END_COLUMN</td>
<td>NVARCHAR(256)</td>
<td>Displays the period end column name.</td>
</tr>
<tr>
<td>HAS_TIMELINE_INDEX</td>
<td>VARCHAR(5)</td>
<td>For internal use only.</td>
</tr>
<tr>
<td>HISTORY_SCHEMA_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the schema of the history table associated to the system-versioned table.</td>
</tr>
<tr>
<td>HISTORY_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the history table associated to the system-versioned table.</td>
</tr>
</tbody>
</table>

Related Information

Temporal Tables
System-Versioned Tables
Application-Time Period Tables
SAP HANA History Tables
TABLES System View [page 1669]
6.1.175  TEXT_CONFIGURATIONS System View

Provides information about configurations and other resources for customizing text analysis and text mining.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema containing the resource.</td>
</tr>
<tr>
<td>NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the resource.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR (16)</td>
<td>Displays the resource type: hdbtextconfig (configuration), textdict (dictionary), textrule (ruleset), textminingconfig, and so on.</td>
</tr>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the name of the resource owner.</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the resource was created or last updated.</td>
</tr>
<tr>
<td>DATA</td>
<td>BLOB</td>
<td>Displays the resource content.</td>
</tr>
</tbody>
</table>

Related Information

Text Mining [page 1455]
Text Mining: TM_CATEGORIZE_KNN Function [page 1456]
Text Mining: TM_GETRELATEDDOCUMENTS Function [page 1460]
Text Mining: TM_GETRELATEDTERMS Function [page 1463]
Text Mining: TM_GETRELEVANTDOCUMENTS Function [page 1467]
Text Mining: TM_GETRELEVANTTERMS Function [page 1470]
Text Mining: TM_GETSUGGESTEDTERMS Function [page 1473]
M_TEXT_ANALYSIS_LANGUAGES System View [page 2243]
M_TEXT_ANALYSIS_MIME_TYPES System View [page 2243]
6.1.176 TIMEZONES System View

Provides information about the timezones which are available together with their originating data set.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMEZONE_NAME</td>
<td>NVARCHAR(64)</td>
<td>Displays the name of the respective timezone that can be used as parameter to give the source respective destination timezone for UTCTOLOCAL and LOCALTOUTC.</td>
</tr>
<tr>
<td>TIMEZONE_DATASET</td>
<td>NVARCHAR(8)</td>
<td>Displays the data set in which the timezones definition is located. It can take the values sap or platform as they are allowed for the dataset parameter to UTCTOLOCAL/LOCALTOUTC. The value of this column can thus be directly used as an input for the respective UTCTOLOCAL/LOCALTOUTC parameter.</td>
</tr>
</tbody>
</table>

**Related Information**

- ALTER SYSTEM CLEAR TIMEZONE CACHE Statement (System Management) [page 523]
- LOCALTOUTC Function (Datetime) [page 249]
- UTCTOLOCAL Function (Datetime) [page 390]
- M_TIMEZONE_ALERTS System View [page 2244]
6.1.177 TRANSACTION_HISTORY System View

Provides information about committed transactions and their users.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMIT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp.</td>
</tr>
<tr>
<td>COMMIT_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the committed transaction.</td>
</tr>
<tr>
<td>APP_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the user that used the transaction.</td>
</tr>
</tbody>
</table>

Related Information

M_TRANSACTIONS System View [page 2250]
M_BLOCKED_TRANSACTIONS System View [page 1761]
M_TRANS_TOKENS System View [page 2253]
M_ES_TRANSACTIONS System View [page 1899]
CURRENT_UPDATE_TRANSACTION Function (Miscellaneous) [page 175]
Transaction Management Statements [page 1208]
SET TRANSACTION Statement (Transaction Management) [page 1173]
SET TRANSACTION AUTOCOMMIT DDL Statement (Transaction Management) [page 1176]
HOST_UNCOMMITTED_WRITE_TRANSACTION View (Embedded Statistics Service)
Blocked Transaction Monitoring
Blocked Transactions
Autonomous Transaction
COMMIT and ROLLBACK
SAVEPOINT
6.1.178 TRIGGERS System View

Provides information about triggers that are defined for tables.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the trigger.</td>
</tr>
<tr>
<td>TRIGGER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the trigger name.</td>
</tr>
<tr>
<td>TRIGGER_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the trigger.</td>
</tr>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the trigger owner.</td>
</tr>
<tr>
<td>OWNER_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the trigger owner.</td>
</tr>
<tr>
<td>SUBJECT_TABLE_SCHEMA</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the table, the trigger is defined for.</td>
</tr>
<tr>
<td>SUBJECT_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of the table, the trigger is defined for.</td>
</tr>
<tr>
<td>TRIGGER_ACTION_TIME</td>
<td>VARCHAR(10)</td>
<td>Displays the time the trigger is executed: BEFORE, AFTER, or INSTEAD OF the specified event.</td>
</tr>
<tr>
<td>TRIGGER_EVENT</td>
<td>VARCHAR(20)</td>
<td>Displays the event the trigger is defined for: combination of DELETE, INSERT, or UPDATE.</td>
</tr>
<tr>
<td>-triggered ACTION_LEVEL</td>
<td>VARCHAR(9)</td>
<td>Displays the level of the event where the triggered action happens: ROW/STATEMENT.</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NCLOB</td>
<td>Displays the query string of the trigger.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the trigger is valid or not: TRUE/FALSE. This becomes FALSE when its base objects are changed or dropped.</td>
</tr>
<tr>
<td>IS_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the trigger is enabled or not: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
### CREATE_TIME

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the trigger was created.</td>
</tr>
</tbody>
</table>

### Related Information

- TRIGGER_ORDERS System View [page 1695]
- CREATE TRIGGER Statement (Data Definition) [page 876]
- DROP TRIGGER Statement (Data Definition) [page 982]

### 6.1.179 TRIGGER_ORDERS System View

Provides information about trigger order for triggers in the database.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the trigger.</td>
</tr>
<tr>
<td>TRIGGER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the trigger name.</td>
</tr>
<tr>
<td>TRIGGER_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the trigger.</td>
</tr>
<tr>
<td>REFERENCED_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the referenced trigger.</td>
</tr>
<tr>
<td>REFERENCED_TRIGGER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the referenced trigger.</td>
</tr>
<tr>
<td>REFERENCED_TRIGGER_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the referenced trigger.</td>
</tr>
<tr>
<td>ORDER_TYPE</td>
<td>VARCHAR(10)</td>
<td>Displays the type of ordering: FOLLOWS/PRECEDES.</td>
</tr>
</tbody>
</table>
Related Information

TRIGGERS System View [page 1694]
CREATE TRIGGER Statement (Data Definition) [page 876]
DROP TRIGGER Statement (Data Definition) [page 982]

6.1.180 USERGROUPS System View

Provides details on all user groups.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERGROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user group.</td>
</tr>
<tr>
<td>USERGROUP_ID</td>
<td>BIGINT</td>
<td>Displays the user group ID.</td>
</tr>
<tr>
<td>CREATOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the creator of the user group.</td>
</tr>
<tr>
<td>IS_USER_ADMIN_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether a user with the USER ADMIN system privilege can manage the specified user group: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_CLIENT_CONNECT_ENABLED</td>
<td>NVARCHAR(5)</td>
<td>Displays whether a user in this user group is able to connect to the database: TRUE/FALSE.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the description for the specified user group.</td>
</tr>
</tbody>
</table>

Additional Information

Users see different values in this view depending on their privileges, as follows:

- Users with one of the following privileges can see everything for all users: USER ADMIN, CATALOG READ, DATA ADMIN
- All other users see only information specific to themselves.
6.1.181 USERGROUP_PARAMETERS System View

Provides the list of parameter sets defined for usergroups.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERGROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the usergroup.</td>
</tr>
<tr>
<td>PARAMETER_SET_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the name of the parameter set the parameter belongs to.</td>
</tr>
<tr>
<td>IS_PARAMETER_SET_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays if the parameter set is enabled for the usergroup: TRUE/FALSE.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the parameter.</td>
</tr>
<tr>
<td>PARAMETER_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the value of the parameter.</td>
</tr>
</tbody>
</table>

Additional Information

Users see different values in this view depending on their privileges, as follows:

- Users with one of the following privileges can see everything for all usergroups: USER ADMIN, CATALOG READ, DATA ADMIN, or USERGROUP OPERATOR
- All other users see only information specific to the usergroup they are in.
6.1.182 USERS System View

Lists all users.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user.</td>
</tr>
<tr>
<td>USER_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the user.</td>
</tr>
<tr>
<td>USERGROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user group that the user belongs to; otherwise, NULL.</td>
</tr>
<tr>
<td>USER_MODE</td>
<td>VARCHAR(8)</td>
<td>Displays the mode of the user: LOCAL/EXTERNAL.</td>
</tr>
<tr>
<td>EXTERNAL_IDENTITY</td>
<td>NVARCHAR(256)</td>
<td>Displays the external identity of the user.</td>
</tr>
<tr>
<td>CREATOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the creator of the user, SYSTEM, or the ID of a user with USER ADMIN privilege. If the user was created by an LDAP provider, the value is SYS.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time.</td>
</tr>
<tr>
<td>VALID_FROM</td>
<td>TIMESTAMP</td>
<td>Displays the start time of the user’s validity.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VALID_UNTIL</td>
<td>TIMESTAMP</td>
<td>Displays the end time of the user’s validity.</td>
</tr>
<tr>
<td>LAST_SUCCESSFUL_CONNECT</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last successful connection of the user.</td>
</tr>
<tr>
<td>LAST_INVALID_CONNECT_ATTEMPT</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last invalid connection attempt.</td>
</tr>
<tr>
<td>INVALID_CONNECT_ATTEMPTS</td>
<td>INTEGER</td>
<td>Displays the number of invalid connection attempts since the last successful connection.</td>
</tr>
<tr>
<td>ADMIN_GIVEN_PASSWORD</td>
<td>VARCHAR(5)</td>
<td>Displays whether the password was provided by the administrator or by the user: TRUE/FALSE.</td>
</tr>
<tr>
<td>LAST_PASSWORD_CHANGE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last password change.</td>
</tr>
<tr>
<td>PASSWORD_CHANGE_NEEDED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the user is forced to change their own password: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_PASSWORD_LIFETIME_CHECK_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the password-lifetime will be checked for the user: TRUE/FALSE.</td>
</tr>
<tr>
<td>USER_DEACTIVATED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the user is deactivated: TRUE/FALSE.</td>
</tr>
<tr>
<td>DEACTIVATION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time given with an explicit deactivation command for the specified user.</td>
</tr>
<tr>
<td>IS_PASSWORD_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether authentication using a password is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_KERBEROS_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether authentication using KERBEROS is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_SAML_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether authentication using SAML is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_X509_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether authentication using an X.509 certificate is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_SAP_LOGON_TICKET_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether authentication using an SAP logon ticket is enabled: TRUE/</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE.</td>
</tr>
<tr>
<td>IS_SAP_ASSERTION_TICKET_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether authentication using an SAP assertion ticket is enabled:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_RESTRICTED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the user is missing the PUBLIC role and privilege on their</td>
</tr>
<tr>
<td></td>
<td></td>
<td>own schema: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_RESTRICTEDDETAILS</td>
<td>VARCHAR(40)</td>
<td>Displays the missing privilege(s): ROLE PUBLIC/CREATE ANY ON OWN SCHEMA.</td>
</tr>
<tr>
<td>IS_CLIENT_CONNECT_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the user is allowed to connect outside of applications:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE/FALSE.</td>
</tr>
<tr>
<td>HAS_REMOTE_USERS</td>
<td>VARCHAR(5)</td>
<td>Displays whether the user has a remote identity mapping: TRUE/FALSE.</td>
</tr>
<tr>
<td>PASSWORD_CHANGE_TIME</td>
<td>TIMESTAMP</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>AUTHORIZATION_MODE</td>
<td>VARCHAR(5)</td>
<td>Displays the Authorization mode of the user: LOCAL/LDAP.</td>
</tr>
<tr>
<td>IS_JWT_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether authentication using JWT is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_LDAP_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether authentication using LDAP is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>CREATE_PROVIDER_TYPE</td>
<td>VARCHAR(16)</td>
<td>If the user was created by a provider using automatic user provisioning, then</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value indicates the type of provider: LDAP PROVIDER/SAML PROVIDER. If the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user was created by another user with USER ADMIN privilege (using a CREATE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USER statement), then the value is NULL.</td>
</tr>
<tr>
<td>CREATE_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the provider that created the user, or NULL if the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user was created by a user with USER ADMIN privilege (using a CREATE USER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>statement).</td>
</tr>
</tbody>
</table>
**Additional Information**

Users see different values in this view depending on their privileges, as follows:

- Users with one of the following privileges can see everything for all users: USER ADMIN, CATALOG READ, DATA ADMIN, USERGROUP OPERATOR
- All other users see only information specific to themselves.

**Related Information**

- CREATE USER Statement (Access Control) [page 893]
- ALTER USER Statement (Access Control) [page 654]
- VALIDATE USER Statement (Access Control) [page 1198]
- DROP USER Statement (Access Control) [page 985]
- USERGROUPS System View [page 1696]
- Database Users
- Manage Users
- Managing SAP HANA Users
- Create a Database User
- Deactivate a Database User
- Delete a Database User
- Change a User
- Reactivate a User
- Deactivate a User
- Delete a User
- User Details
- User Authorization
- The SYSTEM User
- Create a User Group
- User Group Details
6.1.183 USER_PARAMETERS System View

Lists all parameters and their values, which have been assigned to users in the system (using CREATE USER ... SET PARAMETER or ALTER USER ... SET PARAMETER).

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user.</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>NVARCHAR(256)</td>
<td>Displays the client parameter name.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the parameter value.</td>
</tr>
</tbody>
</table>

Related Information

USERS System View [page 1698]
Create User-Specific Parameters
Additional User Parameters
User Self-Service Initialization Parameters
Restrict Use of the CLIENT User Parameter
Maintain User Self-Service Initialization Parameters

6.1.184 VIEWS System View

Lists available views.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view name.</td>
</tr>
<tr>
<td>VIEW_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the view.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_UNICODE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the query string contains Unicode: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_READ_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether this view is read-only or updatable: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_DDL_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether users can query the view and modify the underlying table.</td>
</tr>
<tr>
<td>HAS_CHECK_OPTION</td>
<td>VARCHAR(5)</td>
<td>Displays whether this view has an updatable view condition: TRUE/FALSE.</td>
</tr>
<tr>
<td>HAS_COLUMN_ALIASES</td>
<td>VARCHAR(5)</td>
<td>Displays whether the view has a columns alias: TRUE/FALSE.</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NCLOB</td>
<td>Displays the view definition.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the view description.</td>
</tr>
<tr>
<td>IS_COLUMN_VIEW</td>
<td>VARCHAR(5)</td>
<td>Displays whether this view is a column view: TRUE/FALSE.</td>
</tr>
<tr>
<td>VIEW_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of view: ROW, OLAP, JOIN, HIERARCHY, or CALC.</td>
</tr>
<tr>
<td>IS_TENANT_SHARED_METADATA</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is a global table or not: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the view is valid or not. This value is FALSE when its base objects are changed or dropped: TRUE/FALSE.</td>
</tr>
<tr>
<td>HAS_STRUCTURED_PRIVILEGE_CHECK</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the view is registered for a structured privilege check: TRUE/FALSE.</td>
</tr>
<tr>
<td>HAS_MASKED_COLUMNS</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the view has a mask definition for at least one column: TRUE/FALSE.</td>
</tr>
<tr>
<td>MASK_MODE</td>
<td>NVARCHAR(12)</td>
<td>Displays the mask mode to be applied if the view has masked columns: DEFAULT or SESSION USER; otherwise NULL.</td>
</tr>
<tr>
<td>HAS_PARAMETERS</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the view has parameters defined: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HAS_ANONYMIZATION</td>
<td>VARCHAR(5)</td>
<td>Displays whether the view is defined as anonymized: TRUE/FALSE.</td>
</tr>
<tr>
<td>HAS_CACHE</td>
<td>VARCHAR(36)</td>
<td>Displays whether the result cache is enabled for the view: STATIC, FULL/STATIC, PARTIAL (SELECTED COLUMNS)/STATIC, PARTIAL (FILTERED)/STATIC, PARTIAL (SELECTED COLUMNS, FILTERED)/DYNAMIC, FULL/DYNAMIC, or PARTIAL (FILTERED)/NONE.</td>
</tr>
<tr>
<td>CACHE_RETENTION</td>
<td>INTEGER</td>
<td>Displays the cache refresh interval in minutes.</td>
</tr>
<tr>
<td>CACHE_FILTER</td>
<td>NCLOB</td>
<td>Displays the filter condition specified to reduce the cache size.</td>
</tr>
<tr>
<td>IS_CACHE_FORCED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the result cache is forced or cachability is checked: TRUE/ FALSE.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE VIEW Statement (Data Definition) [page 904]
ALTER VIEW Statement (Data Definition) [page 667]
DROP VIEW Statement (Data Definition) [page 988]
ANONYMIZATION_VIEWS System View [page 1495]
VIEW_COLUMNS System View [page 1705]
VIEW_EXPRESSION_MACROS System View [page 1708]
VIEW_PARAMETERS System View [page 1709]
Systems View
Opening Tables and Views
System and Statistics Views
Monitoring Views for the Backup Catalog
System Views for Monitoring Partitions
Toolbar Options in the Systems View
View Memory Usage

SAP HANA SQL Reference Guide for SAP HANA Platform
System Views Reference
6.1.185 VIEW_COLUMNS System View

Lists available view columns.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view name.</td>
</tr>
<tr>
<td>VIEW_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the view.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view column name.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the view column.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the data type ID.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the data type name.</td>
</tr>
<tr>
<td>OFFSET</td>
<td>SMALLINT</td>
<td>Displays the offset of the column in a record.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the number of chars for character types; max number of digits for numeric types; number of chars for datetime types; and number of bytes for LOB types.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Numeric types: Displays the maximum number of digits to the right of the decimal point. Time, TIMESTAMP: defines the decimal digits as the number of digits to the right of the decimal point in the seconds component of the data.</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is allowed to accept a NULL value: TRUE/FALSE.</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the default value.</td>
</tr>
<tr>
<td>COLLATION</td>
<td>NVARCHAR(256)</td>
<td>Displays the collation.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the description for this column.</td>
</tr>
<tr>
<td>MAX_VALUE</td>
<td>VARCHAR(1)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>MIN_VALUE</td>
<td>VARCHAR(1)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>CS_DATA_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the column store data type ID.</td>
</tr>
<tr>
<td>CS_DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the column store data type name.</td>
</tr>
<tr>
<td>DDIC_DATA_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the DDIC data type ID.</td>
</tr>
<tr>
<td>DDIC_DATA_TYPE_NAME</td>
<td>VARCHAR(7)</td>
<td>Displays the DDIC data type name.</td>
</tr>
<tr>
<td>COMPRESSION_TYPE</td>
<td>VARCHAR(9)</td>
<td>Displays the type of compression: NONE, DEFAULT, PREFIXED, SPARSE, CLUSTERED, INDIRECT or RLE.</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the type of index: NONE/FULL.</td>
</tr>
<tr>
<td>COLUMN_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the column.</td>
</tr>
<tr>
<td>PRELOAD</td>
<td>VARCHAR(5)</td>
<td>Displays if the column is preloaded: TRUE/FALSE.</td>
</tr>
<tr>
<td>GENERATED_ALWAYS_AS</td>
<td>NVARCHAR(1000)</td>
<td>Displays the expression of the column created by GENERATED... AS.</td>
</tr>
<tr>
<td>HAS_SCHEMA_FLEXIBILITY</td>
<td>VARCHAR(5)</td>
<td>Displays if column has schema flexibility: TRUE/FALSE.</td>
</tr>
<tr>
<td>FUZZY_SEARCH_INDEX</td>
<td>VARCHAR(5)</td>
<td>Displays if the column has a fuzzy search index: TRUE/FALSE.</td>
</tr>
<tr>
<td>FUZZY_SEARCH_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the fuzzy search mode.</td>
</tr>
<tr>
<td>MEMORY_THRESHOLD</td>
<td>INTEGER</td>
<td>Displays the memory threshold in bytes for LOB types.</td>
</tr>
<tr>
<td>LOAD_UNIT</td>
<td>VARCHAR(1)</td>
<td>Displays the unit of column data loading: TABLE, COLUMN, or PAGE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GENERATION_TYPE</td>
<td>VARCHAR(1)</td>
<td>Displays ALWAYS AS if the column is a generated column, ALWAYS AS IDENTITY or BY DEFAULT AS IDENTITY if the column is an identity column whose values are always generated or generated by default, and NULL otherwise</td>
</tr>
<tr>
<td>IS_CACHABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is part of cached data: TRUE/FALSE</td>
</tr>
<tr>
<td>IS_CACHE_KEY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is part of the cache key: TRUE/FALSE</td>
</tr>
<tr>
<td>ROW_ORDER_POSITION</td>
<td>VARCHAR(1)</td>
<td>Displays the row order position</td>
</tr>
<tr>
<td>IS_HIDDEN</td>
<td>VARCHAR(5)</td>
<td>Displays whether the specified column is hidden: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_MASKED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is masked: TRUE/FALSE.</td>
</tr>
<tr>
<td>MASK_EXPRESSION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the mask expression (only visible to users with the CATALOG READ, DATA ADMIN, or the SELECT METADATA privilege on the schema, or who own the view or schema).</td>
</tr>
<tr>
<td>CLIENTSIDE_ENCRYPTION_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays whether the column is encrypted (ENCRYPTED) or is in cleartext (NULL).</td>
</tr>
<tr>
<td>CLIENTSIDE_ENCRYPTION_COLUMN_KEY_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the UUID of the column encryption key used to encrypt the column. The value is NULL if the column is not encrypted.</td>
</tr>
<tr>
<td>CLIENTSIDE_ENCRYPTION_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the encryption mode. Possible values are RANDOM, DETERMINISTIC, or NULL (not encrypted).</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY</td>
<td>VARCHAR(1)</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>NUMA_NODE_INDEXES</td>
<td>VARCHAR(1)</td>
<td>Displays the comma-separated list of ranges of user-specified logical NUMA node indexes.</td>
</tr>
</tbody>
</table>
6.1.186  VIEW_EXPRESSION_MACRO System View

Describes the expression macros defined for views.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name for the view.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the view.</td>
</tr>
<tr>
<td>EXPRESSION_MACRO_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the macro.</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the definition (formula) for the macro.</td>
</tr>
</tbody>
</table>
6.1.187 VIEW_PARAMETERS System View

Provides information about view parameters.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the view.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the view.</td>
</tr>
<tr>
<td>VIEW_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the view.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the parameter name.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the data type ID.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the data type name.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the parameter length in bytes.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Displays the scale of the parameter.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the parameter.</td>
</tr>
<tr>
<td>HAS_DEFAULT_VALUE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the parameter has a default value or not: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

- VIEWS System View [page 1702]
- VIEW_COLUMNS System View [page 1705]
- VIEW_EXPRESSION_MACROS System View [page 1708]
- USER_PARAMETERS System View [page 1702]
- CS_VIEW_PARAMETERS System View [page 1536]
- System Properties
- Column View Parameter Binding
### 6.1.188 VIRTUAL_COLUMNS System View

Provides information about virtual columns.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the SQL data type name of the column.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the number of characters for character types, the number of max-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>imum digits for numeric types, the number of characters for datetime types,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or the number of bytes for LOB types.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>For numeric types: displays the maximum number of digits to the right of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the decimal point. For time, TIME-STAMP types: displays the decimal digits</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to the right of the decimal point in the second’s component of the data.</td>
</tr>
<tr>
<td>REMOTE_DATA_TYPE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the remote data type name.</td>
</tr>
<tr>
<td>REMOTE_LENGTH</td>
<td>INTEGER</td>
<td>Displays the remote column length.</td>
</tr>
<tr>
<td>REMOTE_SCALE</td>
<td>INTEGER</td>
<td>Displays the remote scale.</td>
</tr>
<tr>
<td>IS_INSERTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not INSERT is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_UPDATABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not UPDATE is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_UPSERTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not UPSERT is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_SELECTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not SELECT is supported: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
6.1.189  VIRTUAL_COLUMN_PROPERTIES System View

Provides the properties set on virtual table columns.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the virtual table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the virtual table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the virtual table column name.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>NVARCHAR(256)</td>
<td>Displays the property name.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(512)</td>
<td>Displays the property value.</td>
</tr>
<tr>
<td>IS_READ_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the property is read-only: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
6.1.190 VIRTUAL_FUNCTIONS System View

Provides information about virtual functions.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the schema name of the function.</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the name of the function.</td>
</tr>
<tr>
<td>FUNCTION_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the function.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the remote source name used in the function.</td>
</tr>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR (32)</td>
<td>Displays the name of the remote source adapter.</td>
</tr>
<tr>
<td>FUNCTION_CONFIGURATION</td>
<td>NVARCHAR (5000)</td>
<td>Displays the virtual function configuration.</td>
</tr>
<tr>
<td>PACKAGE_SCHEMA_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the package schema name.</td>
</tr>
<tr>
<td>PACKAGE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the package name used by virtual functions.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>NVARCHAR (5)</td>
<td>Displays whether the function is valid or not: TRUE/FALSE. This value be-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>comes FALSE when its base objects are changed or dropped.</td>
</tr>
<tr>
<td>FUNCTION_USAGE_TYPE</td>
<td>NVARCHAR(6)</td>
<td>Displays the usage type of the function: 'SCALAR', 'TABLE', 'AGGREGATE',</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or 'WINDOW'.</td>
</tr>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the owner of the function.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time.</td>
</tr>
</tbody>
</table>

**Related Information**

VIRTUAL_FUNCTION_PACKAGES System View [page 1713]
6.1.191 VIRTUAL_FUNCTION_PACKAGES System View

Provides information about virtual function packages.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the schema name of the virtual function package.</td>
</tr>
<tr>
<td>PACKAGE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the name of the virtual function package.</td>
</tr>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the name of the remote source adapter.</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the virtual function package is created.</td>
</tr>
<tr>
<td>CONTENT</td>
<td>BLOB</td>
<td>Displays the content of the virtual function package.</td>
</tr>
</tbody>
</table>

**Related Information**

VIRTUAL_FUNCTIONS System View [page 1712]
CREATE VIRTUAL FUNCTION Statement (Procedural) [page 917]
SQL Functions [page 83]
## 6.1.192 VIRTUAL_FUNCTION_PARAMETERS System View

Provides information about virtual function parameters.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the function.</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the function.</td>
</tr>
<tr>
<td>FUNCTION_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the function.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the parameter.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the data type ID.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>NVARCHAR(16)</td>
<td>Displays the data type name.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the length of the parameter.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Displays the scale of the parameter.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the parameter.</td>
</tr>
<tr>
<td>TABLE_TYPE_SCHEMA</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of table type if DATA_TYPE_NAME is TABLE_TYPE.</td>
</tr>
<tr>
<td>TABLE_TYPE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of table type if DATA_TYPE_NAME is TABLE_TYPE.</td>
</tr>
<tr>
<td>IS_INPLACE_TYPE</td>
<td>NVARCHAR(5)</td>
<td>Displays whether the parameter type is an inplace type: 'TRUE'/'FALSE'.</td>
</tr>
<tr>
<td>PARAMETER_TYPE</td>
<td>NVARCHAR(7)</td>
<td>Displays the parameter mode: IN, OUT, INOUT.</td>
</tr>
<tr>
<td>HAS_DEFAULT_VALUE</td>
<td>NVARCHAR(5)</td>
<td>Displays whether the parameter has a default value or not: 'TRUE'/'FALSE'.</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>NVARCHAR(5)</td>
<td>Displays whether the parameter accepts a null value: 'TRUE'/'FALSE'.</td>
</tr>
</tbody>
</table>
Related Information

Virtualizing Table User-Defined Functions

6.1.193 VIRTUAL_FUNCTION_PARAMETER_COLUMNS
System View

Provides information about virtual function parameter columns.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the function.</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the function.</td>
</tr>
<tr>
<td>FUNCTION_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the function.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the parameter.</td>
</tr>
<tr>
<td>PARAMETER_POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the parameter.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column in the table parameter.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the column in the table parameter.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>NVARCHAR(16)</td>
<td>Displays the data type name of the column.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the number of chars for char types, number of max digits for numeric types, number of chars for date-time types, number of bytes for LOB types.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Displays the maximum number of digits to the right of the decimal point. For TIME and TIMESTAMP, displays the number of digits to the right of the decimal point in the seconds component of the value.</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>NVARCHAR(5)</td>
<td>Displays whether the parameter accepts a null value: 'TRUE'/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

Virtualizing Table User-Defined Functions

6.1.194 VIRTUAL_PACKAGES System View

Provides the list of virtual packages.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the virtual package.</td>
</tr>
<tr>
<td>PACKAGE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the virtual package.</td>
</tr>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the remote source adapter.</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the virtual package was created.</td>
</tr>
<tr>
<td>CONTENT</td>
<td>BLOB</td>
<td>Displays the content of the virtual package.</td>
</tr>
</tbody>
</table>
Related Information

VIRTUAL_FUNCTION_PACKAGES System View [page 1713]
CREATE VIRTUAL FUNCTION Statement (Procedural) [page 917]
Package Privileges

6.1.195 VIRTUAL_PROCEDURES System View

Provides the list of virtual procedures.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the virtual procedure.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the virtual procedure.</td>
</tr>
<tr>
<td>PROCEDURE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the virtual procedure.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name used in the procedure.</td>
</tr>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the remote source adapter.</td>
</tr>
<tr>
<td>PROCEDURE_CONFIGURATION</td>
<td>NCLOB</td>
<td>Displays the virtual procedure configuration.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the virtual procedure is valid: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

CREATE VIRTUAL PROCEDURE Statement (Procedural) [page 923]
CREATE VIRTUAL PROCEDURE Statement [Smart Data Integration] Procedures
6.1.196 VIRTUAL_TABLES System View

Provides information about virtual tables.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>REMOTE_DB_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote database name.</td>
</tr>
<tr>
<td>REMOTE_OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote owner name.</td>
</tr>
<tr>
<td>REMOTE_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote object name.</td>
</tr>
<tr>
<td>IS_INSERTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not INSERT is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_UPDATABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not UPDATE is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_DELETABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not DELETE is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_UPSERTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not UPSERT is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_SELECTABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not SELECT is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_REMOTE_SUBSCRIPTION_SUPPORTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not REMOTE SUBSCRIPTION is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>ISREMOTE_SUBSCRIPTION_TRANSACTIONAL</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not transactional REMOTE SUBSCRIPTION is supported: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

CREATE VIRTUAL TABLE Statement (Data Definition) [page 925]
ALTER VIRTUAL TABLE Statement (Data Definition) [page 676]
6.1.197 VIRTUAL_TABLE_PARAMETERS System View

Provides a list of parameters of the virtual tables that refer to column views in a remote SAP HANA database.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Displays the schema of the virtual table.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Displays the name of the virtual table.</td>
</tr>
<tr>
<td>PARAMETER_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Displays the name of the parameter.</td>
</tr>
<tr>
<td>IS_MANDATORY</td>
<td>VARCHAR(5)</td>
<td></td>
<td>Displays whether a parameter is mandatory or not: TRUE/FALSE. If TRUE, the value of the parameter should be provided either by setting the default value or through the SQL query. If FALSE, setting the value of the parameter is optional.</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Displays the default value of the parameter. This column is set to an empty string when the parameter has no default value.</td>
</tr>
</tbody>
</table>
Related Information

VIRTUAL_TABLES System View [page 1718]
CREATE VIRTUAL TABLE Statement (Data Definition) [page 925]
ALTER VIRTUAL TABLE Statement (Data Definition) [page 676]
VIRTUAL_TABLE_PROPERTIES System View [page 1720]

Managing Virtual Tables
Create a Virtual Table
Delete a Virtual Table
List All Virtual Tables
List Virtual Tables by Schema
Refresh a Virtual Table
Table Parameter
Any Table Type Parameter
Type and Length Check for Table Parameters

6.1.198 VIRTUAL_TABLE_PROPERTIES System View

Provides the properties set on virtual tables.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the virtual table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the virtual table name.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>NVARCHAR(256)</td>
<td>Displays the property name.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NCLOB</td>
<td>Displays the property value.</td>
</tr>
<tr>
<td>IS_READ_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the property is read-only: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

VIRTUAL_TABLES System View [page 1718]
CREATE VIRTUAL TABLE Statement (Data Definition) [page 925]
ALTER VIRTUAL TABLE Statement (Data Definition) [page 676]
6.1.199 WORKLOAD_CLASSES System View

Provides information about available workload classes.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the workload class.</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>INTEGER</td>
<td>Displays the priority. Zero is the lowest while nine is the highest priority and multiple workload classes can have the same priority. Five is the default.</td>
</tr>
<tr>
<td>STATEMENT_MEMORY_LIMIT</td>
<td>INTEGER</td>
<td>Displays the maximum memory allocation per statement, in gigabytes.</td>
</tr>
<tr>
<td>STATEMENT_THREAD_LIMIT</td>
<td>INTEGER</td>
<td>Displays the maximum number of parallel threads per statement.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_MEMORY_LIMIT</td>
<td>INTEGER</td>
<td>Displays the aggregated memory limit, in gigabytes, that applies to all statements currently being executed within the workload class.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_THREAD_LIMIT</td>
<td>INTEGER</td>
<td>Displays the aggregated thread limit that applies to all statements currently being executed within the workload class.</td>
</tr>
<tr>
<td>STATEMENT_TIMEOUT</td>
<td>INTEGER</td>
<td>Displays the timeout, in seconds, of the statement execution.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WRITE_TRANSACTION_LIFETIME</td>
<td>INTEGER</td>
<td>Displays the lifetime, in minutes, of long-running, uncommitted write trans-</td>
</tr>
<tr>
<td>IDLE_CURSOR_LIFETIME</td>
<td>INTEGER</td>
<td>Displays the lifetime, in minutes, of long-lived cursors.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_REJECT_CPU_THRESHOLD</td>
<td>TINYINT</td>
<td>Displays the threshold value got rejection based on CPU consumption.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_REJECT_MEMORY_THRESHOLD</td>
<td>TINYINT</td>
<td>Displays the threshold value got rejection based on memory consumption.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_QUEUE_CPU_THRESHOLD</td>
<td>TINYINT</td>
<td>Displays the threshold value to queue based on CPU consumption.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_QUEUE_MEMORY_THRESHOLD</td>
<td>TINYINT</td>
<td>Displays the threshold value to queue based on memory consumption.</td>
</tr>
<tr>
<td>IS_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the workload class is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>PARENT_WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the parent workload class.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE WORKLOAD CLASS Statement (Workload Management) [page 929]
ALTER WORKLOAD CLASS Statement (Workload Management) [page 678]
DROP WORKLOAD CLASS Statement (Workload Management) [page 989]
M_WORKLOAD System View [page 2283]
Managing Workload with Workload Classes
Create a Workload Class
Disable or Enable a Workload Class
Edit a Workload Class
Workload Class Examples
Monitor Workload Classes
Import Workload Classes
Export Workload Classes
WORKLOAD_MAPPINGS System View [page 1723]
## 6.1.200 WORKLOAD_MAPPINGS System View

Provides information about available workload mappings.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORKLOAD_MAPPING_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the workload mapping name.</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the workload class name.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>USERGROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user group.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the session variable APPLICA-TIONUSER.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME_WILDCARD</td>
<td>NVARCHAR(1)</td>
<td>Displays the wildcard character that is allowed in the user name.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the session variable APPLICA-TION.</td>
</tr>
<tr>
<td>APPLICATION_NAME_WILDCARD</td>
<td>NVARCHAR(1)</td>
<td>Displays the wildcard character that is used in APPLICATION_NAME.</td>
</tr>
<tr>
<td>CLIENT</td>
<td>NVARCHAR(3)</td>
<td>Displays the session variable CLIENT.</td>
</tr>
<tr>
<td>CLIENT_WILDCARD</td>
<td>NVARCHAR(1)</td>
<td>Displays the wildcard character that is used in CLIENT.</td>
</tr>
<tr>
<td>APPLICATION_COMPONENT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application component.</td>
</tr>
<tr>
<td>APPLICATION_COMPONENT_NAME_WILDCARD</td>
<td>NVARCHAR(1)</td>
<td>Displays the wildcard character that is used in APPLICATION_COMPONENT_NAME.</td>
</tr>
<tr>
<td>APPLICATION_COMPONENT_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the type of application component.</td>
</tr>
<tr>
<td>APPLICATION_COMPONENT_TYPE_WILDCARD</td>
<td>NVARCHAR(1)</td>
<td>Displays the wildcard character that is used in APPLICATION_COMPONENT_TYPE.</td>
</tr>
<tr>
<td>XS_APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name assigned to the session variable XS.APPLICATIONUSER.</td>
</tr>
</tbody>
</table>

SAP HANA SQL Reference Guide for SAP HANA Platform
System Views Reference
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XS_APPLICATION_USER_NAME_WILDCARD</td>
<td>NVARCHAR(1)</td>
<td>Displays the wildcard character that is used for the session variable XS_APPLICATIONUSER.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays the APPLICATIONSOURCE variable.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE_WILDCARD</td>
<td>NVARCHAR(1)</td>
<td>Displays the wild card of the APPLICATIONSOURCE variable.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object.</td>
</tr>
<tr>
<td>IS_ENABLED</td>
<td>NVARCHAR(5)</td>
<td>Displays the workload class is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>NVARCHAR(5)</td>
<td>Displays whether the workload mapping is valid or not. This value is FALSE when its base object is changed or dropped: TRUE/FALSE</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE WORKLOAD MAPPING Statement (Workload Management) [page 934]
ALTER WORKLOAD MAPPING Statement (Workload Management) [page 681]
DROP WORKLOAD MAPPING Statement (Workload Management) [page 990]
M_WORKLOAD System View [page 2283]
WORKLOAD_CLASSES System View [page 1721]
Create a Workload Class Mapping
Managing Workload with Workload Classes
6.1.201 X509_PROVIDERS System View

Lists all of the X.509 providers configured in the SAP HANA database.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the X.509 provider.</td>
</tr>
<tr>
<td>ISSUER_NAME</td>
<td>NVARCHAR(512)</td>
<td>Displays the issuer distinguished name.</td>
</tr>
<tr>
<td>OWNER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the provider’s owner.</td>
</tr>
</tbody>
</table>

Related Information

CREATE X509 PROVIDER (Access Control) [page 937]
ALTER X509 PROVIDER (Access Control) [page 684]
DROP X509 PROVIDER (Access Control) [page 991]
X509_PROVIDER_RULES System View [page 1725]
X509_USER_MAPPINGS System View [page 1726]
X.509 Certificate-Based User Authentication

6.1.202 X509_PROVIDER_RULES System View

Lists all of the matching rules for X.509 providers.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X509_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the X.509 provider.</td>
</tr>
<tr>
<td>POSITION</td>
<td>TINYINT</td>
<td>Displays the order of the rule.</td>
</tr>
</tbody>
</table>

Related Information

CREATE X509 PROVIDER (Access Control) [page 937]
ALTER X509 PROVIDER (Access Control) [page 684]
DROP X509 PROVIDER (Access Control) [page 991]
X509_PROVIDERS System View [page 1725]
X509_USER_MAPPINGS System View [page 1726]
X.509 Certificate-Based User Authentication

6.1.203 X509_USER_MAPPINGS System View

Shows the X.509 certificates that are known for each user.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>SUBJECT_NAME</td>
<td>NVARCHAR(512)</td>
<td>Displays the subject of the X.509 certificate of the user.</td>
</tr>
<tr>
<td>ISSUER_NAME</td>
<td>NVARCHAR(512)</td>
<td>Displays the issuer of the X.509 certificate of the user.</td>
</tr>
<tr>
<td>X509_PROVIDER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the X.509 provider. If the user is mapped without a provider, this column is NULL.</td>
</tr>
</tbody>
</table>

Related Information

CERTIFICATES System View [page 1516]
SAML_PROVIDERS System View [page 1648]
6.1.204  XSA_AUDIT_LOG System View

Provides information about XSA audit records. Database users with the AUDIT ADMIN, AUDIT OPERATOR, or AUDIT READ system privilege can view information in this system view. For all other database users, this view is empty.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the time that the event occurred.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the name of the host where the event occurred.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port number.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the name of the service.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>CLIENT_HOST</td>
<td>NVARCHAR(256)</td>
<td>Displays the IP of the client host.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP of the client application.</td>
</tr>
<tr>
<td>CLIENT_PID</td>
<td>BIGINT</td>
<td>Displays the PID of the client process.</td>
</tr>
<tr>
<td>CLIENT_PORT</td>
<td>INTEGER</td>
<td>Displays the port of the client process.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user that is connected to the database.</td>
</tr>
<tr>
<td>STATEMENT_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user who executed the statement.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application user.</td>
</tr>
<tr>
<td>AUDIT_POLICY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the Audit Policy hit.</td>
</tr>
<tr>
<td>EVENT_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays whether the event was successful or not.</td>
</tr>
<tr>
<td>EVENT_LEVEL</td>
<td>VARCHAR(16)</td>
<td>Displays the severity level of the event.</td>
</tr>
<tr>
<td>EVENT_ACTION</td>
<td>VARCHAR(64)</td>
<td>Displays the action performed by the audit event.</td>
</tr>
<tr>
<td>KEY</td>
<td>NVARCHAR(2000)</td>
<td>Displays the attribute that was changed.</td>
</tr>
<tr>
<td>PREV_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the old value of the attribute.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the new value of the attribute.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the SQL statement that caused the event.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>VARCHAR(5000)</td>
<td>Displays extra information about the event.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the event occurrence at the client side.</td>
</tr>
<tr>
<td>XSA_MESSAGE_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the PI address of the event occurrence.</td>
</tr>
<tr>
<td>XSA_TENANT</td>
<td>VARCHAR(36)</td>
<td>Displays the XSA tenant GUID.</td>
</tr>
<tr>
<td>XSA_UUID</td>
<td>VARCHAR(256)</td>
<td>Displays the unique audit log message ID generated by the audit service.</td>
</tr>
<tr>
<td>XSA_CHANNEL</td>
<td>VARCHAR(16)</td>
<td>Displays the communication protocol that was used when the audit event was triggered.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>XSA_ATTACHMENT_ID</td>
<td>NVARCHAR(256)</td>
<td>Displays the ID of the attachment that triggered the event.</td>
</tr>
<tr>
<td>XSA_ATTACHMENT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the attachment that triggered the event.</td>
</tr>
<tr>
<td>XSA_ORGANIZATION_ID</td>
<td>VARCHAR(36)</td>
<td>Displays the application organization GUID.</td>
</tr>
<tr>
<td>XSA_SPACE_ID</td>
<td>VARCHAR(36)</td>
<td>Displays the application space GUID.</td>
</tr>
<tr>
<td>XSA_INSTANCE_ID</td>
<td>VARCHAR(36)</td>
<td>Displays the GUID of the used auditlog service instance.</td>
</tr>
<tr>
<td>XSA_BINDING_ID</td>
<td>VARCHAR(36)</td>
<td>Displays the application binding GUID in regards to the specific auditlog service instance that is being used.</td>
</tr>
<tr>
<td>XSA_OBJECT</td>
<td>NVARCHAR(5000)</td>
<td>Displays the object containing the accessed personal data.</td>
</tr>
<tr>
<td>XSA_DATA_SUBJECT</td>
<td>NVARCHAR(5000)</td>
<td>Displays the owner of the accessed personal data.</td>
</tr>
</tbody>
</table>

**Related Information**

*Logging and Auditing in XS Advanced*

**6.2 Monitoring Views**

SAP HANA includes a set of runtime views called monitoring views that provide actual SAP HANA runtime data, including statistics and status information related to the execution of DML statements. These views are useful for monitoring and troubleshooting performance. The data in monitoring views is not stored on disk; it is calculated when you execute a query on one of the views.

All monitoring views in SAP HANA start with M_.

6.2.1 M_ABSTRACT_SQL_PLAN_OVERVIEW System View

Provides the status of each Plan Stability Manager on every index server in SAP HANA.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>STATE</td>
<td>VARCHAR(16)</td>
<td>Displays the SQL Plan Stability Manager state:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• READY: plan stability is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CAPTURE: the abstract SQL plans are being captured.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• APPLY: the captured abstract SQL plans are being applied for execution plan generation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• IMPORT: the abstract SQL plans are being imported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MIGRATION: the abstract SQL plans are being migrated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• UPDATELOCATION: the location information of the captured abstract SQL plans is being updated.</td>
</tr>
<tr>
<td>TOTAL_PLAN_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of enabled and disabled loaded plans.</td>
</tr>
<tr>
<td>ENABLED_PLAN_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of enabled plans.</td>
</tr>
<tr>
<td>DISABLED_PLAN_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of disabled plans.</td>
</tr>
<tr>
<td>ALLOCATED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated memory size, in bytes, being used by plan stability.</td>
</tr>
<tr>
<td>LAST_APPLY_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last APPLY start time.</td>
</tr>
<tr>
<td>LAST_APPLY_STOP_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last APPLY stop time.</td>
</tr>
<tr>
<td>LAST_CAPTURE_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last CAPTURE start time.</td>
</tr>
<tr>
<td>LAST_CAPTURE_STOP_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last CAPTURE stop time.</td>
</tr>
</tbody>
</table>
### Related Information

**ABSTRACT_SQL_PLANS System View** [page 1477]

**ALTER SYSTEM [ENABLE | DISABLE | REMOVE] ABSTRACT SQL PLAN** (System Management) [page 532]

**ALTER SYSTEM MIGRATE ABSTRACT SQL PLAN** (System Management) [page 544]

**ALTER SYSTEM [START | STOP] CAPTURE ABSTRACT SQL PLAN** (System Management) [page 575]

**ALTER SYSTEM [START | STOP] APPLY ABSTRACT SQL PLAN** (System Management) [page 570]

**ALTER SYSTEM UPDATE ABSTRACT SQL PLAN Statement** (System Management) [page 586]

**M_ABSTRACT_SQL_PLAN_STATISTICS System View** [page 1731]

SQL Plan Stability

### 6.2.2 M_ABSTRACT_SQL_PLAN_STATISTICS System View

Provides SQL query runtime statistics.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port number.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ABSTRACT_SQL_PLAN_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the abstract SQL plan.</td>
</tr>
<tr>
<td>PLAN_ID</td>
<td>BIGINT</td>
<td>Displays the logical plan ID as a non-negative value.</td>
</tr>
<tr>
<td>PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the preparation time.</td>
</tr>
<tr>
<td>MIN_EXECUTION_TIME_USING_ABSTRACT_SQL_PLAN</td>
<td>BIGINT</td>
<td>Displays the minimum execution time of the plans generated with abstract SQL plans.</td>
</tr>
<tr>
<td>MAX_EXECUTION_TIME_USING_ABSTRACT_SQL_PLAN</td>
<td>BIGINT</td>
<td>Displays the maximum execution time of the plans generated with abstract SQL plans.</td>
</tr>
<tr>
<td>AVG_EXECUTION_TIME_USING_ABSTRACT_SQL_PLAN</td>
<td>BIGINT</td>
<td>Displays the average execution time of the plans generated with abstract SQL plans.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_COUNT_USING_ABSTRACT_SQL_PLAN</td>
<td>BIGINT</td>
<td>Displays the total number of executions of the plans generated with abstract SQL plans.</td>
</tr>
<tr>
<td>MIN_EXECUTION_TIME_NOT_USING_ABSTRACT_SQL_PLAN</td>
<td>BIGINT</td>
<td>Displays the minimum execution time of the plans generated by the current optimizer.</td>
</tr>
<tr>
<td>MAX_EXECUTION_TIME_NOT_USING_ABSTRACT_SQL_PLAN</td>
<td>BIGINT</td>
<td>Displays the maximum execution time of the plans generated by the current optimizer.</td>
</tr>
<tr>
<td>AVG_EXECUTION_TIME_NOT_USING_ABSTRACT_SQL_PLAN</td>
<td>BIGINT</td>
<td>Displays the average execution time of the plans generated by the current optimizer.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_COUNT_NOT_USING_ABSTRACT_SQL_PLAN</td>
<td>BIGINT</td>
<td>Displays the total number of executions of the plans generated by the current optimizer.</td>
</tr>
<tr>
<td>LAST_EXECUTION_TIMESTAMP_USING_ABSTRACT_SQL_PLAN</td>
<td>TIMESTAMP</td>
<td>Displays the statistics update time for the plans generated with abstract SQL plans.</td>
</tr>
<tr>
<td>LAST_EXECUTION_TIMESTAMP_NOT_USING_ABSTRACT_SQL_PLAN</td>
<td>TIMESTAMP</td>
<td>Displays the statistics update time for the plans generated by the current optimizer.</td>
</tr>
</tbody>
</table>
Related Information

ABSTRACT_SQL_PLANS System View [page 1477]
ALTER SYSTEM {ENABLE | DISABLE | REMOVE} ABSTRACT SQL PLAN (System Management) [page 532]
ALTER SYSTEM MIGRATE ABSTRACT SQL PLAN (System Management) [page 544]
ALTER SYSTEM (START | STOP) CAPTURE ABSTRACT SQL PLAN (System Management) [page 575]
ALTER SYSTEM (START | STOP) APPLY ABSTRACT SQL PLAN (System Management) [page 570]
ALTER SYSTEM UPDATE ABSTRACT SQL PLAN Statement (System Management) [page 586]
M_ABSTRACT_SQL_PLAN_OVERVIEW System View [page 1730]

SQL Plan Stability

6.2.3 M_ACTIVE_PROCEDURES System View

Provides statistics of procedure execution.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCEDURE_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the procedure host.</td>
</tr>
<tr>
<td>PROCEDURE_PORT</td>
<td>INTEGER</td>
<td>Displays the procedure internal port.</td>
</tr>
<tr>
<td>PROCEDURE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the stored procedure.</td>
</tr>
<tr>
<td>PROCEDURE_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the procedure connection ID.</td>
</tr>
<tr>
<td>PROCEDURE_TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the procedure transaction ID.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the ID of the statement.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the SQL statement string.</td>
</tr>
<tr>
<td>STATEMENT_PARAMETERS</td>
<td>NCLOB</td>
<td>Displays the statement parameters.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STATEMENT_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the status of the statement:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• EXECUTING: the statement is still running</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• COMPLETED: the statement is completed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• COMPILING: the statement is compiling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ABORTED: the statement was aborted</td>
</tr>
<tr>
<td>STATEMENT_EXECUTION_COUNT</td>
<td>INTEGER</td>
<td>Displays the count of statement execution.</td>
</tr>
<tr>
<td>STATEMENT_DEPTH</td>
<td>INTEGER</td>
<td>Displays the statement depth.</td>
</tr>
<tr>
<td>STATEMENT_COMPILE_TIME</td>
<td>BIGINT</td>
<td>Displays the elapsed time spent compiling the statement.</td>
</tr>
<tr>
<td>STATEMENT_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the elapsed time spent executing the statement.</td>
</tr>
<tr>
<td>STATEMENT_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the statement start time.</td>
</tr>
<tr>
<td>STATEMENT_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the statement end time.</td>
</tr>
<tr>
<td>STATEMENT_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID of the statement.</td>
</tr>
<tr>
<td>STATEMENT_TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction ID of the statement.</td>
</tr>
<tr>
<td>STATEMENT_MATERIALIZATION_TIME</td>
<td>BIGINT</td>
<td>Displays the internal table materialization time.</td>
</tr>
<tr>
<td>STATEMENT_MATERIALIZATION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size, in bytes, of the internal table materialization.</td>
</tr>
<tr>
<td>STATEMENT_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the peak amount of memory, in bytes, used for executing each statement inside a procedure. If you are using a distributed execution, shows the sum of local peak memory for multiple servers.</td>
</tr>
</tbody>
</table>
Related Information

ALTER PROCEDURE Statement (Procedural) [page 463]
CREATE PROCEDURE Statement (Procedural) [page 776]
CREATE VIRTUAL PROCEDURE Statement (Procedural) [page 923]
DROP PROCEDURE Statement (Procedural) [page 963]
M_ACTIVE_STATEMENTS System View [page 1735]
Procedural Statements [page 1204]
PROcedures System View [page 1612]
SAP HANA SQLScript Reference
CREATE PROCEDURE
DROP PROCEDURE
ALTER PROCEDURE
Procedures

6.2.4 M_ACTIVE_STATEMENTS System View

Provides a prepared statements list.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the prepared statement ID.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the MD5 hash value for the statement string.</td>
</tr>
<tr>
<td>START_MVCC_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the internal MVCC timestamp of the transaction start time.</td>
</tr>
<tr>
<td>COMPILED_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the compilation timestamp of the statement.</td>
</tr>
<tr>
<td>STATEMENT_STATUS</td>
<td>VARCHAR(128)</td>
<td>Displays the status of the SQL statement.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the SQL statement.</td>
</tr>
<tr>
<td>ALLOCATED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory peak used for executing this statement. In case of distributed execution it is a sum of the local peak memories of multiple servers.</td>
</tr>
<tr>
<td>USED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Deprecated. This value is the same as ALLOCATED_MEMORY_SIZE.</td>
</tr>
<tr>
<td>PLAN_ID</td>
<td>BIGINT</td>
<td>Displays the logical plan ID.</td>
</tr>
<tr>
<td>LAST_EXECUTED_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the recently executed time of the statement. This timestamp is updated when opening cursors and executing DML/DDL, but not when fetching cursor results or closing cursors.</td>
</tr>
<tr>
<td>LAST_ACTION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the recently performed time of action against the statement. This timestamp is updated when opening cursors, executing DML/DDL, fetching results, and closing cursors.</td>
</tr>
<tr>
<td>RECOMPILE_COUNT</td>
<td>BIGINT</td>
<td>Displays the recompile count.</td>
</tr>
<tr>
<td>EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of executions.</td>
</tr>
<tr>
<td>AVG_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the average statement execution time.</td>
</tr>
<tr>
<td>MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum statement execution time.</td>
</tr>
<tr>
<td>MIN_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum statement execution time.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of the statement execution time.</td>
</tr>
<tr>
<td>AVG_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the average time of the statement execution including communication time with the clients.</td>
</tr>
<tr>
<td>MAX_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the maximum time of the statement execution, including communication time with the clients.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MIN_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the minimum time of the statement execution, including communication time with the clients.</td>
</tr>
<tr>
<td>TOTAL_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the sum of the statement execution time, including communication time with clients.</td>
</tr>
<tr>
<td>AVG_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the average memory size, in bytes, used during each execution.</td>
</tr>
<tr>
<td>MAX_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum memory size, in bytes, used during each execution.</td>
</tr>
<tr>
<td>MIN_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum memory size, in bytes, used during each execution.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the sum of the memory size, in bytes, used during each execution.</td>
</tr>
<tr>
<td>AVG_LOCKWAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the average lock wait time for the statement.</td>
</tr>
<tr>
<td>MAX_LOCKWAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum lock wait time for the statement.</td>
</tr>
<tr>
<td>MIN_LOCKWAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum lock wait time for the statement.</td>
</tr>
<tr>
<td>TOTAL_LOCKWAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total lock wait count for the statement.</td>
</tr>
<tr>
<td>TOTAL_LOCKWAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the accumulated lock wait time for the statement.</td>
</tr>
<tr>
<td>AVG_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of statement preparation.</td>
</tr>
<tr>
<td>MAX_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of statement preparation.</td>
</tr>
<tr>
<td>MIN_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of statement preparation.</td>
</tr>
<tr>
<td>TOTAL_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of statement preparation.</td>
</tr>
<tr>
<td>TOTAL_PREPARATION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total count of statement preparation.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>HAS_HOLDABLE_CURSOR</td>
<td>VARCHAR(5)</td>
<td>Displays the holdable cursor existence.</td>
</tr>
<tr>
<td>CURSOR_TYPE</td>
<td>VARCHAR(18)</td>
<td>Displays the type of cursor.</td>
</tr>
<tr>
<td>PARENT_STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the parent prepared statement ID.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays the application source information.</td>
</tr>
<tr>
<td>STATEMENT_MEMORY_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective statement memory limit.</td>
</tr>
<tr>
<td>STATEMENT_THREAD_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective statement thread limit.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_MEMORY_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective total statement memory limit.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_THREAD_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective total statement thread limit.</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the effective workload class used for the execution.</td>
</tr>
</tbody>
</table>

**Additional Information**

Setting the value of the APPLICATION_SOURCE is only available via internal APIs of the SAP HANA database client interfaces (for more information see SAP Note 2873396).

**Related Information**

- SAP Note 2873396
- Monitor and Analyze Active Statements
- M_ACTIVE_PROCEDURES System View [page 1733]
- WORKLOAD_CLASSES System View [page 1721]
6.2.5 M_ADMISSION_CONTROL_EVENTS System View

Displays information about significant events.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the event.</td>
</tr>
<tr>
<td>EVENT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of event.</td>
</tr>
<tr>
<td>EVENT_REASON</td>
<td>VARCHAR(64)</td>
<td>Displays the reason for the event.</td>
</tr>
<tr>
<td>QUEUE_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the queue wait time, in microseconds.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the session connection ID.</td>
</tr>
<tr>
<td>MESSAGE_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the request type.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the prepared statement ID.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the unique identifier for an SQL string.</td>
</tr>
<tr>
<td>CPU_USAGE_RATIO</td>
<td>INTEGER</td>
<td>Displays the measured value of the CPU usage.</td>
</tr>
<tr>
<td>MEMORY_RATIO</td>
<td>INTEGER</td>
<td>Displays the measured memory size, in bytes, as a percentage of the global allocation limit.</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the effective workload class.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
</tbody>
</table>

**Related Information**

- [M_ADMISSION_CONTROL_QUEUES System View](#) [page 1740]
- [M_ADMISSION_CONTROL_STATISTICS System View](#) [page 1741]
- Configuring Admission Control
- Managing Peak Load (Admission Control)
6.2.6 M_ADMISSION_CONTROL_QUEUES System View

Provides detailed information regarding queued session requests by Session-Wise Admission Control.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESSAGE_ID</td>
<td>BIGINT</td>
<td>Displays the request ID of the client packet.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the session connection ID.</td>
</tr>
<tr>
<td>MESSAGE_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the request type.</td>
</tr>
<tr>
<td>STATEMENT_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the statement type (for example, DDL or DML).</td>
</tr>
<tr>
<td>ENQUEUE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the request queue time in microseconds.</td>
</tr>
<tr>
<td>TIMEOUT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the estimated time, in microseconds, when the request should be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>timed-out.</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the effective workload class</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
</tbody>
</table>

Related Information

M_ADMISSION_CONTROL_EVENTS System View [page 1739]
M_ADMISSION_CONTROL_STATISTICS System View [page 1741]
Configuring Admission Control
Managing Peak Load (Admission Control)
Manage Admission Control
# 6.2.7 M_ADMISSION_CONTROL_STATISTICS System View

Provides the overall statistics values of the Session-Wise Admission Control feature.

## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL_ADMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated request admission count.</td>
</tr>
<tr>
<td>TOTAL_REJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated request rejection count.</td>
</tr>
<tr>
<td>TOTAL_ENQUEUE_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated request queued count.</td>
</tr>
<tr>
<td>TOTAL_DEQUEUE_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated request dequeued count (the executed request count).</td>
</tr>
<tr>
<td>TOTAL_TIMEOUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated request dequeued count due to timeout (the rejected request count).</td>
</tr>
<tr>
<td>CURRENT_QUEUE_SIZE</td>
<td>BIGINT</td>
<td>Displays the current waiting request queued count.</td>
</tr>
<tr>
<td>LAST_CPU_USAGE_RATIO</td>
<td>INTEGER</td>
<td>Displays the last measured value of the CPU usage.</td>
</tr>
<tr>
<td>LAST_CPU_USAGE_MEASURE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time at which the last CPU usage ratio was measured.</td>
</tr>
<tr>
<td>LAST_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the last measured memory size in GB.</td>
</tr>
<tr>
<td>LAST_MEMORY_RATIO</td>
<td>BIGINT</td>
<td>Displays the last measured memory size, as a percentage of the global allocation limit (last_memory_size/memory_allocation_limit).</td>
</tr>
<tr>
<td>LAST_MEMORY_MEASURE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time at which the last memory size was measured.</td>
</tr>
<tr>
<td>LAST_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the last wait time of the request in the queue in microseconds.</td>
</tr>
<tr>
<td>AVG_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the average wait time of the request in the queue in microseconds.</td>
</tr>
<tr>
<td>MAX_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum wait time of the request in the queue in microseconds.</td>
</tr>
</tbody>
</table>
### Related Information

- M_ADMISSION_CONTROL_EVENTS System View [page 1739]
- M_ADMISSION_CONTROL_QUEUES System View [page 1740]
- Configuring Admission Control
- Managing Peak Load (Admission Control)
- Manage Admission Control

### 6.2.8 M_AFL_FUNCTIONS System View

Provides application function execution information.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the area of the function.</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the function.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of executions.</td>
</tr>
<tr>
<td>LAST_EXECUTION_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of the last execution.</td>
</tr>
</tbody>
</table>
6.2.9  M_AFL_STATES System View

Provides information about AFL states.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE_ID</td>
<td>NVARCHAR(256)</td>
<td>Displays the ID of the state.</td>
</tr>
<tr>
<td>AREA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the area.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>NVARCHAR(256)</td>
<td>Displays the state description.</td>
</tr>
<tr>
<td>CREATE_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the time when the state was created.</td>
</tr>
<tr>
<td>LAST_ACCESS_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the time stamp of the last access.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
</tbody>
</table>

Related Information

M_AFL_FUNCTIONS System View [page 1742]
AFL_FUNCTIONS System View [page 1484]
AFL_FUNCTION_PARAMETERS System View [page 1485]
AFL_FUNCTION_PROPERTIES System View [page 1486]
AFL_PACKAGES System View [page 1487]
AFL_TEXTS System View [page 1488]
6.2.10 M_AGENTS System View

Provides agent host information.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGENT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the agent name.</td>
</tr>
<tr>
<td>FREE_PHYSICAL_MEMORY</td>
<td>BIGINT</td>
<td>Displays the free physical memory on the agent host in bytes.</td>
</tr>
<tr>
<td>FREE_SWAP_SPACE</td>
<td>BIGINT</td>
<td>Displays the free swap memory on the agent host in bytes.</td>
</tr>
<tr>
<td>LAST_CONNECT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last successful connection time.</td>
</tr>
<tr>
<td>SYS_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the agent host timestamp in the agent host’s local timezone.</td>
</tr>
<tr>
<td>USED_PHYSICAL_MEMORY</td>
<td>BIGINT</td>
<td>Displays the used physical memory on the agent host in bytes.</td>
</tr>
<tr>
<td>USED_SWAP_SPACE</td>
<td>BIGINT</td>
<td>Displays the used swap memory on the agent host in bytes.</td>
</tr>
<tr>
<td>UTC_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the agent host timestamp in UTC.</td>
</tr>
<tr>
<td>AGENT_VERSION</td>
<td>VARCHAR(32)</td>
<td>Displays the agent version.</td>
</tr>
<tr>
<td>AGENT_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the agent status.</td>
</tr>
</tbody>
</table>

Related Information

AGENTS System View [page 1488]
AGENT_CONFIGURATION System View [page 1489]
AGENT_GROUPS System View [page 1490]
Add Hosts Using SAP Host Agent
Using SAP Host Agent to Execute Platform LCM Tasks
6.2.11 M_ANONYMIZATION_VIEWS System View

Provides runtime information about anonymized views.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the anonymized view.</td>
</tr>
<tr>
<td>ANONYMIZATION_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the status of the anonymized view.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for views anonymized using the DIFFERENTIAL_PRIVACY al-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gorithm are READY, INVALID, or REFRESHING.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Possible values for views anonymized using the K-ANONYMITY and L-DIVERSITY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>algorithm are CREATED, REFRESHING, READY, or INVALID.</td>
</tr>
<tr>
<td>REFRESH_RECOMMENDED</td>
<td>VARCHAR(5)</td>
<td>Displays whether a refresh is recommended: TRUE/FALSE.</td>
</tr>
<tr>
<td>LAST_REFRESH_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the finish time of the latest refresh.</td>
</tr>
<tr>
<td>LAST_REFRESH_PARAMETERS</td>
<td>NCLOB</td>
<td>Displays the parameters specified during the last REFRESH VIEW command on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the view, in JSON format.</td>
</tr>
</tbody>
</table>

Related Information

ANONYMIZATION_VIEWS System View [page 1495]
ANONYMIZATION_VIEW_COLUMNS System View [page 1496]
VIEWS System View [page 1702]
CREATE VIEW Statement (Data Definition) [page 904]
ALTER VIEW Statement (Data Definition) [page 667]
REFRESH VIEW Statement (Data Definition) [page 1086]
6.2.12 M_ASYNCronous_TABLE_REPLICAS System View

Provides detailed information about asynchronous table replicas. Deprecated.

This view has a resettable counterpart. This means you can also see the values since the last reset in the dependent view M_ASYNCRONOUS_TABLE_REPLICAS_RESET. To reset the values, use the SQL command: ALTER SYSTEM RESET MONITORING VIEW SYS.M_ASYNCRONOUS_TABLE_REPLICAS_RESET

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Host name</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Internal port</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Persistence Volume ID</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Schema name of replica (replication target)</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Table name of replica (replication target)</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>VARCHAR(6)</td>
<td>Replica table type: ROW, COLUMN</td>
</tr>
<tr>
<td>SOURCE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Schema name of replication source</td>
</tr>
<tr>
<td>SOURCE_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Table name of replication source</td>
</tr>
<tr>
<td>SOURCE_TABLE_TYPE</td>
<td>VARCHAR(6)</td>
<td>Source Table type: ROW, COLUMN</td>
</tr>
<tr>
<td>SOURCE_TABLE_VOLUME_ID</td>
<td>INTEGER</td>
<td>Volume ID of its source table</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Partition ID. 0 for non-partitioned tables, 1 through number of partitions for partitioned tables</td>
</tr>
<tr>
<td>REPLICATION_STATUS</td>
<td>VARCHAR(8)</td>
<td>Current asynchronous replication status: ENABLED, ENABLING, DISABLED</td>
</tr>
<tr>
<td>LAST_ENABLE_TIME</td>
<td>TIMESTAMP</td>
<td>Time of last replication enable</td>
</tr>
<tr>
<td>LAST_DISABLE_TIME</td>
<td>TIMESTAMP</td>
<td>Time of last replication disable</td>
</tr>
<tr>
<td>LAST_ERROR_CODE</td>
<td>INTEGER</td>
<td>Last error code</td>
</tr>
<tr>
<td>LAST_ERROR_MESSAGE</td>
<td>NVARCHAR(5000)</td>
<td>Last error message</td>
</tr>
</tbody>
</table>
### 6.2.13 MASYNCROUS_TABLE_REPLICAS_RESET System View

Shows aggregated information on caches (since last reset).

This view contains values accumulated since the last reset of the main view MASYNCROUS_TABLE_REPLICAS. Refer to MASYNCROUS_TABLE_REPLICAS for information about the structure and use of this view. In addition to the members mentioned in MASYNCROUS_TABLE_REPLICAS, this view also contains a timestamp field RESET_TIME, which indicates the last time the data was reset.

### 6.2.14 MATTACHED_STORAGE System View

Provides information about currently attached devices.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host.</td>
</tr>
<tr>
<td>STORAGE_ID</td>
<td>INTEGER</td>
<td>Displays the storage ID.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the path for the currently attached devices.</td>
</tr>
</tbody>
</table>
### Related Information

- Storage Replication
- SAP HANA Native Storage Extension
- Columnar and Row-Based Data Storage
- Statistics for Page-Loadable Storage
- Persistent Data Storage in the SAP HANA Database

### 6.2.15 M_BACKUP_CATALOG System View

Provides common data for all backup catalog entries.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of the backup catalog entry.</td>
</tr>
<tr>
<td>ENTRY_TYPE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the classification of backup catalog entries. The following types are supported: complete data backup, data snapshot, log backup, log missing, differential, and incremental backups.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BACKUP_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of a data backup or a log backup, respectively. All backup files of a single data backup share the same BACKUP_ID.</td>
</tr>
<tr>
<td>SYS_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time given in the server's local time.</td>
</tr>
<tr>
<td>UTC_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time given in UTC.</td>
</tr>
<tr>
<td>SYS_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the stop time given in the server's local time.</td>
</tr>
<tr>
<td>UTC_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the stop time given in UTC.</td>
</tr>
<tr>
<td>STATE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the result of the corresponding action: successful, failed, running, cancel pending, or canceled.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>VARCHAR(256)</td>
<td>Displays any additional information.</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>VARCHAR(512)</td>
<td>Displays any additional information.</td>
</tr>
<tr>
<td>SYSTEM_ID</td>
<td>VARCHAR(3)</td>
<td>Displays the system identifier (SID) of the SAP HANA database.</td>
</tr>
<tr>
<td>ENCRYPTION_ROOT_KEY_HASH</td>
<td>VARCHAR(64)</td>
<td>Displays the key hash used, if any, to locate the encryption root key in the keystore.</td>
</tr>
<tr>
<td>SOURCE_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the database that created the backup.</td>
</tr>
<tr>
<td>ES_ENCRYPTION_CHANGE_ENABLED</td>
<td>VARCHAR(5)</td>
<td>This column applies to SAP HANA dynamic tiering only. Displays if the backup can be used to change the volume encryption key during recovery. Set to TRUE if DISABLE ES ENCRIPTION CHANGE is shown at backup time and if extended storage is in the topology. Displays NULL if extended storage does not exist in the topology.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RETAINED</td>
<td>VARCHAR(5)</td>
<td>For complete data backups: Indicates whether a backup is flagged as RETAINED. Complete data backups that are flagged as RETAINED cannot be deleted by SAP HANA, either individually, by scheduled housekeeping tasks, or by the SQL statement <code>BACKUP CATALOG DELETE</code>.</td>
</tr>
</tbody>
</table>

**Related Information**

M_BACKUP_CATALOG_FILES System View [page 1751]
BACKUP CATALOG DELETE Statement (Backup and Recovery) [page 690]
Backup Catalog
What Information is in the Backup Catalog?
Monitoring Views for the Backup Catalog
Rebuilding the Backup Catalog
Accumulated Backups of the Backup Catalog
Destination for Backups of the Backup Catalog
Parameters for Backing Up the Backup Catalog
Create Data Backups
SAP HANA Backup
M_BACKUP_CONFIGURATION System View [page 1754]
M_BACKUP_HISTORY_BROKEN System View [page 1756]
M_BACKUP_SIZE_ESTIMATIONS System View [page 1759]
M_BACKUP_PROGRESS System View [page 1758]
RECOVER DATABASE Statement (Backup and Recovery) [page 1070]
RECOVER DATA Statement (Backup and Recovery) [page 1064]
6.2.16 M_BACKUP_CATALOG_FILES System View

Provides location information for all backup catalog entries.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of a backup catalog entry.</td>
</tr>
<tr>
<td>BACKUP_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of a data backup and log backup respectively. All backup files of a single data backup share the same backup ID.</td>
</tr>
<tr>
<td>SOURCE_ID</td>
<td>BIGINT</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>SOURCE_TYPE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the type of persistence to be backed up: volume or topology.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>SERVICE_TYPE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the type of database service: indexserver, nameserver, or statistics-server.</td>
</tr>
<tr>
<td>REDO_LOG_POSITION</td>
<td>BIGINT</td>
<td>In the case of a data backup, this value displays the log position that must be processed next when a log recovery is requested after restoring the data backup.</td>
</tr>
<tr>
<td>FIRST_REDO_LOG_POSITION</td>
<td>BIGINT</td>
<td>In the case of a log backup, this value displays the log position of the oldest log entry contained in the backup.</td>
</tr>
<tr>
<td>LAST_REDO_LOG_POSITION</td>
<td>BIGINT</td>
<td>In the case of a log backup, this value displays the log position of the newest log entry contained in the backup.</td>
</tr>
<tr>
<td>BACKUP_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the backup in bytes.</td>
</tr>
<tr>
<td>DESTINATION_PATH</td>
<td>VARCHAR(512)</td>
<td>Displays that the data or log backup was written to this location.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DESTINATION_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the type of location: file or backint.</td>
</tr>
<tr>
<td>EXTERNAL_BACKUP_ID</td>
<td>VARCHAR(64)</td>
<td>Displays the identifier of the backup received from a backup tool.</td>
</tr>
<tr>
<td>LOG_SEGMENT_COUNT</td>
<td>INTEGER</td>
<td>For a log backup, this value displays the number of log segments contained in the backup.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For complete data backups and delta backups, this column always contains the SQL NULL value.</td>
</tr>
<tr>
<td>BACKUP_CATALOG_BACKUP_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of the backup of the backup catalog that was triggered after a particular data or log backup.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This ID indicates which backup of the backup catalog is the first one that contains that backup operation.</td>
</tr>
<tr>
<td>UTC_LAST_COMMIT_TIME</td>
<td>INTEGER</td>
<td>Displays the most recent commit time in UTC for log backups only.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This time shows the most recent point in time, to which this backup can be used to recover the database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The times of a backup’s start and completion do not indicate the point in time that can be reached in a recovery using that backup. This is because the most recent database commit is older.</td>
</tr>
<tr>
<td>BACKINT_FALLBACK_USED</td>
<td>VARCHAR</td>
<td>For third-party backup tools: Indicates whether a log backup was written as configured, or as a result of an automatic log backup fallback.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE: A log backup was written to the file system as configured.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE: A log backup was originally configured to be written to a third-party backup tool, but was written to the file system due to an automatic log backup fallback.</td>
</tr>
</tbody>
</table>

SAP HANA SQL Reference Guide for SAP HANA Platform
System Views Reference
1752 PUBLIC
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIGURATION_INCLUDED</td>
<td>VARCHAR(5)</td>
<td>FALSE: A data backup contains no user configuration files (.ini files).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE: A data backup contains user configuration files (.ini files).</td>
</tr>
<tr>
<td>COMPRESSED_SIZE</td>
<td>BIGINT</td>
<td>Specifies the payload size after backup compression.</td>
</tr>
<tr>
<td>COMPRESSION_ALGORITHM</td>
<td>VARCHAR(32)</td>
<td>Specifies the name of the compression algorithm used for backup compression.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_BACKUP_CATALOG System View [page 1748]
- BACKUP CATALOG DELETE Statement (Backup and Recovery) [page 690]
- Backup Catalog
- What Information is in the Backup Catalog?
- Monitoring Views for the Backup Catalog
- Rebuilding the Backup Catalog
- Accumulated Backups of the Backup Catalog
- Destination for Backups of the Backup Catalog
- Parameters for Backing Up the Backup Catalog
- Create Data Backups
- SAP HANA Backup
- M_BACKUP_CONFIGURATION System View [page 1754]
- M_BACKUP_HISTORY_BROKEN System View [page 1756]
- M_BACKUP_SIZE_ESTIMATIONS System View [page 1759]
- M_BACKUP_PROGRESS System View [page 1758]
- RECOVER DATABASE Statement (Backup and Recovery) [page 1070]
- RECOVER DATA Statement (Backup and Recovery) [page 1064]
6.2.17 M_BACKUP_CONFIGURATION System View

Provides backup configuration statistics.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX_RECOVERY_FILE_AGE</td>
<td>BIGINT</td>
<td>Displays the maximum recovery file age in seconds.</td>
</tr>
<tr>
<td>LOG_REPLAY_STEP_SIZE</td>
<td>BIGINT</td>
<td>Displays the log replay step size in bytes.</td>
</tr>
<tr>
<td>MAX_RECOVERY_BACKINT_CHANNELS</td>
<td>BIGINT</td>
<td>Displays the maximum number of parallel backint channels per request during recovery.</td>
</tr>
<tr>
<td>BACKINT_EXECUTABLE_LINK</td>
<td>VARCHAR(256)</td>
<td>Displays the backint executable link name.</td>
</tr>
<tr>
<td>BACKINT_EXECUTABLE</td>
<td>VARCHAR(256)</td>
<td>Displays the backint executable file name.</td>
</tr>
<tr>
<td>BACKINT_DATA_BACKUP_PATH</td>
<td>VARCHAR(256)</td>
<td>Displays the data backup directory for the backint.</td>
</tr>
<tr>
<td>BACKINT_LOG_BACKUP_PATH</td>
<td>VARCHAR(256)</td>
<td>Displays the log backup directory for the backint.</td>
</tr>
<tr>
<td>BACKINT_CATALOG_BACKUP_PATH</td>
<td>VARCHAR(256)</td>
<td>Displays the directory for backint-based catalog backups.</td>
</tr>
<tr>
<td>FILE_DATA_BACKUP_PATH</td>
<td>VARCHAR(256)</td>
<td>Displays the default directory for file-based data backups.</td>
</tr>
<tr>
<td>FILE_LOG_BACKUP_PATH</td>
<td>VARCHAR(256)</td>
<td>Displays the current directory for file-based log backups.</td>
</tr>
<tr>
<td>FILE_CATALOG_BACKUP_PATH</td>
<td>VARCHAR(256)</td>
<td>Displays the directory for file-based catalog backups.</td>
</tr>
<tr>
<td>FILE_ROOTKEY_BACKUP_PATH</td>
<td>VARCHAR(256)</td>
<td>Displays the directory for file-based rootkey backups.</td>
</tr>
<tr>
<td>LOG_BACKUP_TIMEOUT</td>
<td>BIGINT</td>
<td>Displays the log backup timeout in seconds.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOG_BACKUP_INTERVAL_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the log backup interval mode.</td>
</tr>
<tr>
<td>IS_ROOT_KEY_BACKUP_PASWORD_SET</td>
<td>VARCHAR(5)</td>
<td>Displays if the root key backup password is set for the specified database: TRUE/FALSE.</td>
</tr>
<tr>
<td>ROOT_KEY_ENCRYPTION_CONTROL</td>
<td>VARCHAR(16)</td>
<td>Displays whether encryption is configured at the system or local database level: SYSTEM DATABASE/LOCAL DATABASE.</td>
</tr>
<tr>
<td>ROOT_KEY_BACKUP_PASWORD_LAST_CHANGE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays when the root key backup password was last changed.</td>
</tr>
<tr>
<td>IS_LOG_BACKUP_COMPRESSION_ENABLED</td>
<td>VARCHAR(5)</td>
<td>TRUE: Backup compression is enabled for log backups.</td>
</tr>
<tr>
<td>LOG_BACKUP_COMPRESSION_ALGORITHM</td>
<td>VARCHAR(32)</td>
<td>Specifies the compression algorithm to be used for log backups.</td>
</tr>
<tr>
<td>LOG_BACKUP_COMPRESSION_LEVEL</td>
<td>INTEGER</td>
<td>Specifies the compression level to be used by the configured algorithm for log backups.</td>
</tr>
<tr>
<td>DATA_BACKUP_COMPRESSION_ALGORITHM</td>
<td>VARCHAR(32)</td>
<td>Specifies the compression algorithm to be used for data backups.</td>
</tr>
<tr>
<td>DATA_BACKUP_COMPRESSION_LEVEL</td>
<td>INTEGER</td>
<td>Specifies the compression level to be used by the configured algorithm for data backups.</td>
</tr>
<tr>
<td>INCLUDE_CONFIGURATION_BACKUP</td>
<td>VARCHAR(5)</td>
<td>TRUE: Backup of customer-specific configuration is enabled. Changed .ini files are included in data backups.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_BACKUP_CATALOG System View [page 1748]
- M_BACKUP_CATALOG_FILES System View [page 1751]
- BACKUP CATALOG DELETE Statement (Backup and Recovery) [page 690]
- Backup Catalog
  - What Information is in the Backup Catalog?
  - Monitoring Views for the Backup Catalog
  - Rebuilding the Backup Catalog
Accumulated Backups of the Backup Catalog
Destination for Backups of the Backup Catalog
Parameters for Backing Up the Backup Catalog
Create Data Backups
SAP HANA Backup
M_BACKUP_HISTORY_BROKEN System View [page 1756]
M_BACKUP_SIZE_ESTIMATIONS System View [page 1759]
M_BACKUP_PROGRESS System View [page 1758]
RECOVER DATABASE Statement (Backup and Recovery) [page 1070]
RECOVER DATA Statement (Backup and Recovery) [page 1064]

6.2.18 M_BACKUP_HISTORY_BROKEN System View

Provides information about broken backup history entries.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of the broken backup history entry.</td>
</tr>
<tr>
<td>SYS_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time at which the broken backup history was detected. The time is provided in the system's local time.</td>
</tr>
<tr>
<td>UTC_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time at which the broken backup history was detected. The time is given in UTC time.</td>
</tr>
<tr>
<td>SOURCE_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>SERVICE_TYPE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the type of database service.</td>
</tr>
<tr>
<td>REDO_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the redo log position of the broken backup history entry.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REASON</td>
<td>VARCHAR(64)</td>
<td>Displays the reason for the broken backup history:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Unknown reason</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Redo log writing disabled on persistence session</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Redo log flush partially turned OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Redo log flush turned ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Migration to “level 2 delta”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• General reason for persistence factory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Extended storage persistence encryption key changed</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>VARCHAR(256)</td>
<td>Displays any additional information.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_BACKUP_CATALOG System View [page 1748]
- M_BACKUP_CATALOG_FILES System View [page 1751]
- BACKUP CATALOG DELETE Statement (Backup and Recovery) [page 690]
- Backup Catalog
- What Information is in the Backup Catalog?
- Monitoring Views for the Backup Catalog
- Rebuilding the Backup Catalog
- Accumulated Backups of the Backup Catalog
- Destination for Backups of the Backup Catalog
- Parameters for Backing Up the Backup Catalog
- Create Data Backups
- SAP HANA Backup
- M_BACKUP_CONFIGURATION System View [page 1754]
- M_BACKUP_SIZE_ESTIMATIONS System View [page 1759]
- M_BACKUP_PROGRESS System View [page 1758]
- RECOVER DATABASE Statement (Backup and Recovery) [page 1070]
- RECOVER DATA Statement (Backup and Recovery) [page 1064]
6.2.19  M_BACKUP_PROGRESS System View

Provides the progress of the most recent backup.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of a data backup. All backup files of a single data backup share the same BACKUP_ID.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the name of the service.</td>
</tr>
<tr>
<td>ENTRY_TYPE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the classification of the type of backup.</td>
</tr>
<tr>
<td>SYS_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the local server time that the backup started.</td>
</tr>
<tr>
<td>UTC_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the backup started.</td>
</tr>
<tr>
<td>SYS_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the local server time that the backup was terminated.</td>
</tr>
<tr>
<td>UTC_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the backup was terminated.</td>
</tr>
<tr>
<td>STATE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the current state of the backup.</td>
</tr>
<tr>
<td>BACKUP_SIZE</td>
<td>BIGINT</td>
<td>Displays the total amount of data in bytes.</td>
</tr>
<tr>
<td>TRANSFERRED_SIZE</td>
<td>BIGINT</td>
<td>Displays the amount of data transferred in bytes.</td>
</tr>
</tbody>
</table>

Related Information

M_BACKUP_CATALOG System View [page 1748]
M_BACKUP_CATALOG_FILES System View [page 1751]
6.2.20  M_BACKUP_SIZE_ESTIMATIONS System View

Provides the estimated size of the next data backup.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>NVARCHAR(32)</td>
<td>Displays the name of the service.</td>
</tr>
<tr>
<td>ENTRY_TYPE_NAME</td>
<td>NVARCHAR(64)</td>
<td>Displays the classification of the type of backup. The following types are supported: complete data backup, differential, and incremental backups.</td>
</tr>
<tr>
<td>ESTIMATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the estimated size of the backup in bytes.</td>
</tr>
</tbody>
</table>
Additional Information

i Note

The actual backup size may differ from the estimation shown in this view.

The ENTRY_TYPE_NAME column is equivalent to the column in the M_BACKUP_CATALOG view. The table contains records for each service with a volume for all the supported backup types. You cannot create a differential or incremental backup or view estimation of a differential or incremental backup if a complete backup does not exist. However, once a complete backup exists, you can view estimations for differential or incremental backups, even if these backups have not been created.

Related Information

M_BACKUP_CATALOG System View [page 1748]
M_BACKUP_CATALOG_FILES System View [page 1751]
BACKUP CATALOG DELETE Statement (Backup and Recovery) [page 690]
Backup Catalog
What Information is in the Backup Catalog?
Monitoring Views for the Backup Catalog
Rebuilding the Backup Catalog
Accumulated Backups of the Backup Catalog
Destination for Backups of the Backup Catalog
Parameters for Backing Up the Backup Catalog
Create Data Backups
SAP HANA Backup
M_BACKUP_CONFIGURATION System View [page 1754]
M_BACKUP_HISTORY_BROKEN System View [page 1756]
M_BACKUP_PROGRESS System View [page 1758]
RECOVER DATABASE Statement (Backup and Recovery) [page 1070]
RECOVER DATA Statement (Backup and Recovery) [page 1064]
### 6.2.21 M_BLOCKED_TRANSACTIONS System View

Provides a transaction list waiting for locks.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>BLOCKED_TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction object ID of the transaction waiting for a lock.</td>
</tr>
<tr>
<td>BLOCKED_UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the write transaction ID of the write transaction waiting for the lock.</td>
</tr>
<tr>
<td>BLOCKED_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID associated with the blocked write transaction.</td>
</tr>
<tr>
<td>LOCK_OWNER_TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction object ID of the transaction holding the lock.</td>
</tr>
<tr>
<td>LOCK_OWNER_UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the write transaction ID of the write transaction holding the lock.</td>
</tr>
<tr>
<td>LOCK_OWNER_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID associated with the write transaction holding the lock.</td>
</tr>
<tr>
<td>BLOCKED_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the blocked timestamp.</td>
</tr>
<tr>
<td>WAITING_RECORD_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the ID of the record on which the lock is currently placed.</td>
</tr>
<tr>
<td>WAITING_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema on which the lock is currently placed.</td>
</tr>
<tr>
<td>WAITING_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Deprecated: use WAITING_OBJECT_NAME instead.</td>
</tr>
<tr>
<td>WAITING_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object on which the lock is currently placed.</td>
</tr>
<tr>
<td>WAITING_OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type.</td>
</tr>
</tbody>
</table>
### Related Information

- **M_TRANSACTIONS System View** [page 2250]
- **TRANSACTION_HISTORY System View** [page 1693]
- **SET TRANSACTION AUTOCOMMIT DDL Statement (Transaction Management)** [page 1176]
- Blocked Transaction Monitoring
- Blocked Transactions
- Autonomous Transaction

### 6.2.22 M_BUFFER_CACHE_POOL_STATISTICS System View

Provides statistics for each buffer pool in a cache.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the disk I/O reads over time.</td>
</tr>
<tr>
<td>MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the cache miss count of the cache group.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume identifier for the index server.</td>
</tr>
<tr>
<td>CACHE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the cache name: C/S, R/S.</td>
</tr>
<tr>
<td>BUFFER_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the buffer in the pool in bytes.</td>
</tr>
<tr>
<td>REPLACEMENT_POLICY</td>
<td>VARCHAR(16)</td>
<td>Displays the type of replacement policy used by the cache.</td>
</tr>
<tr>
<td>GROWTH_PERCENT</td>
<td>INTEGER</td>
<td>Displays the rate, as a percentage, at which the buffer pool can grow:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOTAL_BUFFER_COUNT = TOTAL_BUFFER_COUNT + GROWTH_PERCENT / 100 * MAX_SIZE</td>
</tr>
<tr>
<td>TOTAL BUFFER_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of buffers allocated to the pool.</td>
</tr>
<tr>
<td>FREE BUFFER_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of free buffers for the pool.</td>
</tr>
<tr>
<td>LRU_LIST_BUFFER_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of buffers in the LRU chain for the pool.</td>
</tr>
<tr>
<td>HOT BUFFER_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of buffers in the hot buffer list for the pool.</td>
</tr>
<tr>
<td>BUFFER_REUSE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of buffers released from the LRU list for the pool so</td>
</tr>
<tr>
<td></td>
<td></td>
<td>that a requested page can be cached.</td>
</tr>
<tr>
<td>OUT OF BUFFER_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times that an out-of-buffer situation occurred while</td>
</tr>
<tr>
<td></td>
<td></td>
<td>requesting buffers from the pool.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_BUFFER_CACHE_POOL_STATISTICS_RESET System View [page 1764]
- M_BUFFER_CACHE_STATISTICS System View [page 1764]
- M_BUFFER_CACHE_STATISTICS_RESET System View [page 1765]
- M_LOG BUFFERS System View [page 1995]
6.2.23 M_BUFFER_CACHE_POOL_STATISTICS_RESET System View

Provides statistics for each buffer pool in a cache since the last reset.

This view contains values accumulated since the last reset of the main view M_BUFFER_CACHE_POOL_STATISTICS. Refer to M_BUFFER_CACHE_POOL_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_BUFFER_CACHE_POOL_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_BUFFER_CACHE_POOL_STATISTICS System View [page 1762]

6.2.24 M_BUFFER_CACHE_STATISTICS System View

Provides a cache level overview of the configuration, cache status, and memory usage.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the disk I/O reads over time.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume identifier for the index server.</td>
</tr>
<tr>
<td>CACHE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the cache name: C/S or R/S.</td>
</tr>
<tr>
<td>STATE</td>
<td>VARCHAR(16)</td>
<td>Displays the cache state: ENABLED, DISABLING, DISABLED, or SHRINKING.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REPLACEMENT_POLICY</td>
<td>VARCHAR(16)</td>
<td>Displays the type of replacement policy used by the cache.</td>
</tr>
<tr>
<td>MAX_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum buffer cache memory capacity in bytes.</td>
</tr>
<tr>
<td>MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the cache miss count of the cache group.</td>
</tr>
<tr>
<td>ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated memory for the buffer cache in bytes.</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the used memory in buffer cache in bytes.</td>
</tr>
<tr>
<td>BUFFER_REUSE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times that a buffer is released for reuse by the cache.</td>
</tr>
<tr>
<td>HIT_RATIO</td>
<td>FLOAT</td>
<td>Displays the ratio of pages found in the buffer cache to pages requested from the buffer cache.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_BUFFER_CACHE_POOL_STATISTICS System View [page 1762]
- M_BUFFER_CACHE_POOL_STATISTICS_RESET System View [page 1764]
- M_BUFFER_CACHE_STATISTICS_RESET System View [page 1765]
- M_LOG_BUFFERS System View [page 1995]
- SAP HANA NSE Buffer Cache

**6.2.25 M_BUFFER_CACHE_STATISTICS_RESET System View**

Provides a cache level overview of the configuration, cache status, and memory usage since the last reset.

This view contains values accumulated since the last reset of the main view M_BUFFER_CACHE_STATISTICS. Refer to M_BUFFER_CACHE_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_BUFFER_CACHE_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.
Related Information

M_BUFFER_CACHE_STATISTICS System View [page 1764]

6.2.26 M_CACHES System View

Provides aggregated information on caches.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>CACHE_ID</td>
<td>VARCHAR(128)</td>
<td>Displays the unique identifier for the cache.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum available memory budget in bytes available for the cache instance.</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory in bytes used by the cache instance.</td>
</tr>
<tr>
<td>ENTRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of entries in the cache instance.</td>
</tr>
<tr>
<td>INSERT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of insertions into the cache instance.</td>
</tr>
<tr>
<td>INVALIDATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of invalidations in the cache instance.</td>
</tr>
<tr>
<td>HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of cache hits for the cache instance.</td>
</tr>
<tr>
<td>MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of cache misses for the cache instance.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LAST_ACCESS_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last access of the cache instance.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view has a resettable counterpart; you can see the values since the last reset in the M_CACHES_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_CACHES_RESET;
```

**Related Information**

- [ALTER SYSTEM CLEAR CACHE Statement (System Management) [page 518]](#)
- [ALTER SYSTEM REMOVE RESULT CACHE ENTRY Statement (System Management) [page 561]](#)
- [ALTER SYSTEM CLEAR SQL PLAN CACHE Statement (System Management) [page 522]](#)
- [ALTER SYSTEM REFRESH RESULT CACHE ENTRY Statement (System Management) [page 558]](#)
- [ALTER SYSTEM REMOVE SQL PLAN CACHE ENTRY Statement (System Management) [page 562]](#)
- [ALTER SYSTEM CLEAR TIMEZONE CACHE Statement (System Management) [page 523]](#)
- [ALTER SYSTEM CLEAR RESULT CACHE Statement (System Management) [page 522]](#)
- [ALTER SYSTEM REFRESH RESULT CACHE Statement (System Management) [page 557]](#)
- [ALTER SYSTEM RECOMPILE SQL PLAN CACHE ENTRY Statement (System Management) [page 555]](#)
- [ALTER SYSTEM (PIN | UNPIN) SQL PLAN CACHE ENTRY Statement (System Management) [page 548]](#)
- [M_DYNAMIC_RESULT_CACHE System View [page 1875]](#)
- [M_SQL_PLAN_CACHE_PARAMETERS System View [page 2160]](#)
- [M_CACHE_ENTRIES System View [page 1768]](#)

**6.2.27 M_CACHES_RESET System View**

Provides aggregated information on caches since the last reset.

This view contains values accumulated since the last reset of the main view M_CACHES. Refer to M_CACHES for information about the structure and use of this view.

In addition to the members mentioned in M_CACHES, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.
6.2.28 M_CACHE_ENTRIES System View

Provides cache entry information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>CACHE_ID</td>
<td>VARCHAR(128)</td>
<td>Displays the ID of the cache that created the entry.</td>
</tr>
<tr>
<td>ENTRY_ID</td>
<td>VARCHAR(128)</td>
<td>Displays the ID of the cached entry.</td>
</tr>
<tr>
<td>ENTRY_DESCRIPTION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the description of the cached entry.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VARCHAR(128)</td>
<td>Displays the information about the component that created the cached entry.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the information about the user that created the cached entry.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the used memory to store the cached entry in the cache in bytes.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the point in time when the cache entry was inserted into the cache in bytes.</td>
</tr>
<tr>
<td>READ_COUNT</td>
<td>BIGINT</td>
<td>Determines how often the cache entry was read successfully from the cache.</td>
</tr>
<tr>
<td>LAST_ACCESS_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last access of the cache instance.</td>
</tr>
</tbody>
</table>
6.2.29 M_CATALOG_MEMORY System View

Provides memory usage information by catalog manager.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>VARCHAR(128)</td>
<td>Displays the category of the catalog data.</td>
</tr>
<tr>
<td>ALLOCATION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of allocated entries.</td>
</tr>
<tr>
<td>ALLOCATED_FIXED_PART_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated memory size in bytes for the fixed-size part.</td>
</tr>
<tr>
<td>USED_FIXED_PART_SIZE</td>
<td>BIGINT</td>
<td>Displays the used memory size in bytes for the fixed-size part.</td>
</tr>
<tr>
<td>ALLOCATED_VARIABLE_PART_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated memory size in bytes for the variable-size part.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>USED_VARIABLE_PART_SIZE</td>
<td>BIGINT</td>
<td>Displays the used memory size in bytes for the variable-size part.</td>
</tr>
</tbody>
</table>

**Related Information**

*M_MEMORY System View* [page 2007]
Memory Usage in the SAP HANA Database
Memory Management
Persistent Memory
Memory Sizing

### 6.2.30 M_CE_CALCSCENARIOS System View

Provides all available calculation scenarios.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>SCENARIO_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the scenario name.</td>
</tr>
<tr>
<td>IS_PERSISTENT</td>
<td>VARCHAR(5)</td>
<td>Indicates if the calculation scenario is persistent or transient: TRUE or FALSE.</td>
</tr>
<tr>
<td>FLAGS</td>
<td>BIGINT</td>
<td>Displays the calculation scenario flags.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size of the loaded calculation scenario model in bytes.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>NVARCHAR(256)</td>
<td>Displays the component that created the calculation scenario.</td>
</tr>
</tbody>
</table>
### Related Information

- **M_CE_CALCSCEENARIOS_OVERVIEW** System View [page 1771]
- **M_CE_CALSCENARIO_HINTS** System View [page 1772]
- **M_CE_CALCVIEW_DEPENDENCIES** System View [page 1772]

### 6.2.31 M_CE_CALSCENARIOS_OVERVIEW System View

Provides an overview of Calcscenarios without JSON representation.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCENARIO_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the scenario name.</td>
</tr>
<tr>
<td>IS_PERSISTENT</td>
<td>VARCHAR(5)</td>
<td>Indicates if the calculation scenario is persistent or transient: TRUE/FALSE.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size of the loaded calculation scenario model in bytes.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>NVARCHAR(256)</td>
<td>Displays the component that create the calculation scenario.</td>
</tr>
</tbody>
</table>
Related Information

M_CE_CALCSCENARIOS System View [page 1770]
M_CE_CALCSCENARIO_HINTS System View [page 1772]
M_CE_CALCVIEW_DEPENDENCIES System View [page 1772]

6.2.32 M_CE_CALCSCENARIO_HINTS System View

Exposes all hints that are defined in a calculation scenario.

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>SCENARIO_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the scenario name.</td>
</tr>
<tr>
<td>HINT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the hint name. For example, qo_pop_hints.</td>
</tr>
<tr>
<td>HINT_VALUE</td>
<td>NVARCHAR(256)</td>
<td>Displays the hint value. For example, USE_OCAP_PLAN.</td>
</tr>
</tbody>
</table>

Related Information

M_CE_CALCSCENARIOS System View [page 1770]
M_CE_CALCSCENARIOS_OVERVIEW System View [page 1771]
M_CE_CALCVIEW_DEPENDENCIES System View [page 1772]

6.2.33 M_CE_CALCVIEW_DEPENDENCIES System View

Provides all views that are referencing a CalculationScenario.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the column view.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column view name.</td>
</tr>
<tr>
<td>CALCNODE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the referenced calculation node name.</td>
</tr>
<tr>
<td>SCENARIO_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the scenario name.</td>
</tr>
</tbody>
</table>

**Related Information**

M_CE_CALCCENARIOS System View [page 1770]
M_CE_CALCCENARIOS_OVERVIEW System View [page 1771]
M_CE_CALCCENARIO_HINTS System View [page 1772]

### 6.2.34 M_CE_DEBUG_INFOS System View

Provides debug information after the execution of a calculation scenario.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>SCENARIO_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the scenario name.</td>
</tr>
<tr>
<td>NODE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the calculation node name.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>BIGINT</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>DEBUG_TYPE_MASK</td>
<td>BIGINT</td>
<td>Displays the bit mask indicating the type of debug information.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name (temporary intermediate result table).</td>
</tr>
<tr>
<td>COLUMN_NAMES</td>
<td>CLOB</td>
<td>Displays the column names.</td>
</tr>
<tr>
<td>TABLE_SIZE</td>
<td>BIGINT</td>
<td>Displays the table size in bytes (temporary intermediate result table).</td>
</tr>
<tr>
<td>EXECUTION_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the execution timestamp.</td>
</tr>
<tr>
<td>APPLICATION_QUERY_ID</td>
<td>NVARCHAR(256)</td>
<td>Displays the application query ID.</td>
</tr>
</tbody>
</table>

**Related Information**

- **M_CE_DEBUG_JSONS System View** [page 1774]
- **M_CE_DEBUG_NODE_MAPPING System View** [page 1775]
- **M_DEBUG_SESSIONS System View** [page 1865]
- **M_DEBUG_CONNECTIONS System View** [page 1864]
- SQLScript Debugger

**6.2.35 M_CE_DEBUG_JSONS System View**

Provides all available JSONS (original, instantiated, or optimized) of a scenario for a concrete query.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCENARIO_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the scenario name.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>BIGINT</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the JSON type: original, instantiated, optimized, or extrace.</td>
</tr>
</tbody>
</table>
## Related Information

- **M_CE_DEBUG_INFOS System View** [page 1773]
- **M_CE_DEBUG_NODE_MAPPING System View** [page 1775]
- **M_DEBUG_SESSIONS System View** [page 1865]
- **M_DEBUG_CONNECTIONS System View** [page 1864]

### SQLScript Debugger

6.2.36 **M_CE_DEBUG_NODE_MAPPING System View**

Provides information about node mapping between calculation nodes and Runtime nodes after execution.

## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>BIGINT</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>SCENARIO_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the scenario name.</td>
</tr>
<tr>
<td>NODE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the calculation node name.</td>
</tr>
<tr>
<td>NODE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the calculation node type (ROOT, TEMPLATE, or CALC_DS).</td>
</tr>
<tr>
<td>SUCC_SCENARIO_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the successor scenario name.</td>
</tr>
<tr>
<td>SUCC_NODE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the successor calculation node name.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
RUNTIME_NODE_NAME | NVARCHAR(256) | Displays the runtime calculation node name.
RUNTIME_NODE_JSON | NCLOB | Displays the JSON string representing the calculation node.
EXECUTION_TIMESTAMP | TIMESTAMP | Displays the execution timestamp.

**Related Information**

M_CE_DEBUG_INFOS System View [page 1773]
M_CE_DEBUG_JSONS System View [page 1774]
M_DEBUG_SESSIONS System View [page 1865]
M_DEBUG_CONNECTIONS System View [page 1864]
SQLScript Debugger

### 6.2.37 M_CE_PLE_CALCSCENARIOS System View

Provides all available calculation scenarios created by the PlanningEngine.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of calculation scenario.</td>
</tr>
<tr>
<td>SCENARIO_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the calculation scenario name.</td>
</tr>
</tbody>
</table>

**Related Information**

M_CE_CALCSCENARIOS System View [page 1770]
M_CE_CALCSCENARIOS_OVERVIEW System View [page 1771]
6.2.38  M_CLIENT_VERSIONS System View

Provides versions of all supported client applications.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIENT_TYPE</td>
<td>VARCHAR(256)</td>
<td>Displays the type of client, for example, ABAP_FDA or BI Modeler.</td>
</tr>
<tr>
<td>CLIENT_RELEASE_ID</td>
<td>SMALLINT</td>
<td>Displays the technical release ID.</td>
</tr>
<tr>
<td>CLIENT_RELEASE_DESC</td>
<td>VARCHAR(256)</td>
<td>Displays the human readable release description.</td>
</tr>
<tr>
<td>MIN_VERSION</td>
<td>SMALLINT</td>
<td>Displays the lowest supported protocol version.</td>
</tr>
<tr>
<td>MAX_VERSION</td>
<td>SMALLINT</td>
<td>Displays the highest supported protocol version.</td>
</tr>
</tbody>
</table>

**Related Information**

Client Certificates
Configuring Clients for Failover
Connections from Database Clients and Web Clients to SAP HANA
6.2.39 M_COLLECTION_TABLES System View

Provides information about JSON collections.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>COLLECTION_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the collection.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time of the table collection.</td>
</tr>
<tr>
<td>ESTIMATED_DOCUMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the estimated document count.</td>
</tr>
<tr>
<td>ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated size of the collection in bytes.</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of the collection in bytes.</td>
</tr>
<tr>
<td>LOAD_STATUS</td>
<td>VARCHAR</td>
<td>Displays whether the table collection is LOADED or UNLOADED.</td>
</tr>
</tbody>
</table>

Related Information

M_COLLECTION_TABLEVIRTUAL_FILES System View [page 1779]
DROP COLLECTION Statement (JSON Document Store) [page 952]
The JSON Document Store
6.2.40 M_COLLECTION_TABLE_VIRTUAL_FILES System View

Provides information about the virtual files for JSON collections.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>COLLECTION_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the collection.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID.</td>
</tr>
<tr>
<td>FILE_TYPE</td>
<td>VARCHAR</td>
<td>Displays the type of file.</td>
</tr>
<tr>
<td>INTERNAL_OBJECT_ID</td>
<td>BIGINT</td>
<td>Displays the internal identifier for the table.</td>
</tr>
<tr>
<td>FILE</td>
<td>VARCHAR(256)</td>
<td>Displays the file.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID.</td>
</tr>
<tr>
<td>PHYSICAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the storage size used for the file in bytes.</td>
</tr>
<tr>
<td>PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of storage pages used for the file.</td>
</tr>
</tbody>
</table>

Related Information

M_COLLECTION_TABLES System View [page 1778]
The JSON Document Store
DROP COLLECTION Statement (JSON Document Store) [page 952]
## 6.2.41 M_COMPACTION_THREAD System View

Provides compaction thread statistics.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>NUM_COMPACTION_COLLISIONS</td>
<td>BIGINT</td>
<td>Displays the count of memory compaction collisions, which includes other threads currently in compaction and unforced compactions.</td>
</tr>
<tr>
<td>NUM_COMPACTIONS</td>
<td>BIGINT</td>
<td>Displays the number of compaction requests.</td>
</tr>
<tr>
<td>LAST_SIZE_COMPACTION_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the last compaction request.</td>
</tr>
<tr>
<td>MAX_SIZE_COMPACTION_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the maximum sized compaction request in bytes.</td>
</tr>
<tr>
<td>MIN_SIZE_COMPACTION_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the minimum sized compaction request in bytes.</td>
</tr>
<tr>
<td>SUM_SIZE_COMPACTION_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the total size of compaction requests in bytes.</td>
</tr>
<tr>
<td>AVG_SIZE_COMPACTION_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the average size of compaction requests in bytes.</td>
</tr>
<tr>
<td>LAST_SIZE_FREEABLE</td>
<td>BIGINT</td>
<td>Displays the size in bytes compacted by the last at compaction call.</td>
</tr>
<tr>
<td>MAX_SIZE_FREEABLE</td>
<td>BIGINT</td>
<td>Displays the size in bytes compacted by the maximum compaction call.</td>
</tr>
<tr>
<td>MIN_SIZE_FREEABLE</td>
<td>BIGINT</td>
<td>Displays the size in bytes compacted by the minimum compaction call.</td>
</tr>
<tr>
<td>SUM_SIZE_FREEABLE</td>
<td>BIGINT</td>
<td>Displays the size in bytes compacted by all compaction calls (total).</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AVG_SIZE_FREEABLE</td>
<td>BIGINT</td>
<td>Displays the size in bytes compacted by the average compaction call.</td>
</tr>
<tr>
<td>LAST_SIZE_FREED_BY_GARBAGE_COLLECTION</td>
<td>BIGINT</td>
<td>Displays the size in bytes compacted by the last memory garbage collection (defragmentation).</td>
</tr>
<tr>
<td>MAX_SIZE_FREED_BY_GARBAGE_COLLECTION</td>
<td>BIGINT</td>
<td>Displays the size in bytes compacted by the largest memory garbage collection (defragmentation).</td>
</tr>
<tr>
<td>MIN_SIZE_FREED_BY_GARBAGE_COLLECTION</td>
<td>BIGINT</td>
<td>Displays the size in bytes compacted by the smallest memory garbage collection (defragmentation).</td>
</tr>
<tr>
<td>SUM_SIZE_FREED_BY_GARBAGE_COLLECTION</td>
<td>BIGINT</td>
<td>Displays the size in bytes compacted by all memory garbage collections (defragmentation) (total).</td>
</tr>
<tr>
<td>AVG_SIZE_FREED_BY_GARBAGE_COLLECTION</td>
<td>BIGINT</td>
<td>Displays the size in bytes compacted by the average memory garbage collection (defragmentation).</td>
</tr>
<tr>
<td>LAST_COMPACTION_RESULT</td>
<td>BIGINT</td>
<td>Displays the last compaction result in bytes. This is the difference of allocated bytes before and after compaction and may be influenced by other factors than compaction.</td>
</tr>
<tr>
<td>MAX_COMPACTION_RESULT</td>
<td>BIGINT</td>
<td>Displays the largest compaction result in bytes. This is the difference of allocated bytes before and after compaction and may be influenced by other factors than compaction.</td>
</tr>
<tr>
<td>MIN_COMPACTION_RESULT</td>
<td>BIGINT</td>
<td>Displays the smallest compaction result in bytes. This is the difference of allocated bytes before and after compaction and may be influenced by other factors than compaction.</td>
</tr>
<tr>
<td>SUM_COMPACTION_RESULT</td>
<td>BIGINT</td>
<td>Displays the total of all compaction results in bytes. This is the difference of allocated bytes before and after compaction and may be influenced by other factors than compaction.</td>
</tr>
</tbody>
</table>
Additional Information

The compaction thread automatically calls for memory reduction (including for the resource container) if memory gets low, but is not already exhausted. The compaction thread also executes the memory reduction requests triggered by other processes (inter-process memory management). This view displays some information about the compaction requests and the compacted sizes.

Related Information

Thread Monitoring
Thread Details
Threads

6.2.42 M_CONDITIONAL_VARIABLES System View

Provides semaphore statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>STATISTICS_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the statistics object name.</td>
</tr>
<tr>
<td>STATISTICS_ID</td>
<td>BIGINT</td>
<td>Displays the statistics object unique ID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of wait calls.</td>
</tr>
<tr>
<td>BLOCKING_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of blocking wait calls.</td>
</tr>
<tr>
<td>TIMEOUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of timeouts.</td>
</tr>
<tr>
<td>WAIT_RATE</td>
<td>DOUBLE</td>
<td>Displays the wait rate percentage.</td>
</tr>
<tr>
<td>LAST_BLOCKING_TIME</td>
<td>BIGINT</td>
<td>Displays the latest time of blocking wait calls in microseconds.</td>
</tr>
<tr>
<td>MAX_BLOCKING_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of blocking wait calls in microseconds.</td>
</tr>
<tr>
<td>MIN_BLOCKING_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of blocking wait calls in microseconds.</td>
</tr>
<tr>
<td>SUM_BLOCKING_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of blocking wait calls in microseconds.</td>
</tr>
<tr>
<td>AVG_BLOCKING_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of blocking wait calls in microseconds.</td>
</tr>
<tr>
<td>CREATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of semaphore creations (for shared statistics only).</td>
</tr>
<tr>
<td>DESTROY_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of semaphore destructsions (for shared statistics only).</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VARCHAR(32)</td>
<td>Displays the component.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view contains information about single conditional variable objects or groups of conditional variable objects. It does not contain information about all conditional variables.

**Related Information**

- [M_CONDITIONAL_VARIABLES_RESET System View](#) [page 1784]
- Conditional Breakpoints
- Session Variables [page 75]
- Conditional Breakpoints
- Conditionals
6.2.43  M_CONDITIONAL_VARIABLES_RESET System View

Semaphore statistics (since last reset).

This view contains values accumulated since the last reset of the main view M_CONDITIONAL_VARIABLES. Refer to M_CONDITIONAL_VARIABLES for information about the structure and use of this view. In addition to the members mentioned in M_CONDITIONAL_VARIABLES, this view also contains a timestamp field RESET_TIME, which indicates the last time the data was reset.

Related Information

M_CONDITIONAL_VARIABLES System View [page 1782]

6.2.44  M_CONFIGURATION_PARAMETER_VALUES System View

Displays landscape service parameter values.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SECTION</td>
<td>VARCHAR(128)</td>
<td>Displays the section of the parameter.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>KEY</td>
<td>VARCHAR(128)</td>
<td>Displays the parameter key.</td>
</tr>
<tr>
<td>HAS_KEY_INDEX</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the parameter is an indexed parameter: TRUE/FALSE.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the name of the file where the parameter value is defined.</td>
</tr>
<tr>
<td>LAYER_NAME</td>
<td>VARCHAR(8)</td>
<td>Displays the layer that the parameter is defined on.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the parameter value.</td>
</tr>
<tr>
<td>HAS_PROPERTIES</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the parameter has properties: TRUE or FALSE.</td>
</tr>
<tr>
<td>RAW_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the raw string value of the parameter as it is cached in the service.</td>
</tr>
<tr>
<td>VIOLATED_RESTRICIONS</td>
<td>VARCHAR(64)</td>
<td>Displays the restrictions violated by the set value: CUSTOM, VALUE_RESTRICTION, or LAYER_RESTRICTION.</td>
</tr>
<tr>
<td>RESTART_REQUIRED</td>
<td>VARCHAR(5)</td>
<td>Displays if the parameter value has been changed and requires a restart to become effective: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

- [CONFIGURATION_PARAMETER_PROPERTIES System View](page 1521)
- [Backup Configuration Parameters](page 1521)
- [Database-Specific Configuration Parameters](page 1521)
- [Configuration Parameters for the Table Consistency Check](page 1521)
- [ALTER SYSTEM ALTER CONFIGURATION Statement [Dynamic Tiering]](page 1312)
- [ALTER SYSTEM ALTER CONFIGURATION Statement (System Management)](page 500)
- [ALTER SYSTEM ALTER CONFIGURATION Statement [Accelerator for SAP ASE]](page 1285)

**6.2.45 M_CONNECTIONS System View**

Provides detailed information on connections between a client and a database. Information includes: connection status, client information, connection type, and resource utilization.
## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction object ID.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the connected time.</td>
</tr>
<tr>
<td>IDLE_TIME</td>
<td>BIGINT</td>
<td>Displays the time that the connection is unused and idle in milliseconds.</td>
</tr>
<tr>
<td>CONNECTION_STATUS</td>
<td>VARCHAR(128)</td>
<td>Displays the connection status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>RUNNING</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A statement is executing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>IDLE</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No statements are currently executing on this connection.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>QUEUEING</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The connection is currently queued. The status changes to RUNNING when it is dequeued (this depends on the system’s resource consumption).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>EMPTY</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A historic connection that is removed after 1 hour. The removal time can be configured in <code>indexserver.ini</code> [session] connection_history_life time = 60 # minutes.</td>
</tr>
<tr>
<td>CLIENT_HOST</td>
<td>NVARCHAR(256)</td>
<td>Displays the host name of client machine.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP of client machine.</td>
</tr>
<tr>
<td>CLIENT_PID</td>
<td>BIGINT</td>
<td>Displays the client process ID.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CONNECTION_TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the connection type. Connection types include: Remote, Local, History (remote), and History (local).</td>
</tr>
<tr>
<td>OWN</td>
<td>VARCHAR(5)</td>
<td>Displays the user's own connection. Returns TRUE if it is your own connection and FALSE if it is not.</td>
</tr>
<tr>
<td>IS_HISTORY_SAVED</td>
<td>VARCHAR(5)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>MEMORY_SIZE_PER_CONNECTION</td>
<td>BIGINT</td>
<td>Displays the allocated memory size per connection in bytes.</td>
</tr>
<tr>
<td>AUTO_COMMIT</td>
<td>VARCHAR(5)</td>
<td>Displays the commit mode of the current transaction. Returns TRUE if the current connection is in auto-commit mode and FALSE if it is not.</td>
</tr>
<tr>
<td>LAST_ACTION</td>
<td>VARCHAR(128)</td>
<td>Displays the last action completed by the current connection. Actions include: ExecuteGroup, CommitTrans, AbortTrans, PrepareStatement, CloseStatement, ExecutePrepared, ExecuteStatement, FetchCursor, CloseCursor, LobGetPiece, LogPutPiece, LobFind, Authenticate, Connect, Disconnect, ExecQidItab, CursorFetchItab, InsertIncompleteItab, AbapStream, TxStartXA, and TxJoinXA.</td>
</tr>
<tr>
<td>CURRENT_STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the current statement ID.</td>
</tr>
<tr>
<td>CURRENT_OPERATOR_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the current operator name.</td>
</tr>
<tr>
<td>FETCHED_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the sum of the record count fetched by select statements.</td>
</tr>
<tr>
<td>AFFECTED_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the sum of the record count affected by DML/DDL statements, for example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INSERT/UPDATE/DELETE/REPLACE/CREATE TABLE ... LIKE ... WITH DATA/CREATE TABLE ... AS (SELECT ..)/IMPORT/EXPORT</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SENT_MESSAGE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of messages sent by the current connection in bytes.</td>
</tr>
<tr>
<td>SENT_MESSAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total message count sent by the current connection.</td>
</tr>
<tr>
<td>RECEIVED_MESSAGE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of messages received by the current connection in bytes.</td>
</tr>
<tr>
<td>RECEIVED_MESSAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total message count received by the current connection.</td>
</tr>
<tr>
<td>CREATOR_THREAD_ID</td>
<td>BIGINT</td>
<td>Displays the thread ID that created the current connection.</td>
</tr>
<tr>
<td>CREATED_BY</td>
<td>VARCHAR(256)</td>
<td>Displays the engine component that created the connections. These include: Session, Planning, Repository, CalcEngine, Authentication, Table Exporter, Loader, LLVM, JSVM, IMS Search API, OLAP Engine, Mergedog, Ping Status, Name Server, Queue Server, SQL Stored Procedure, Authorization, Trex-ViaDbsl from ABAP, HybridTable Reorganizer, and Session external.</td>
</tr>
<tr>
<td>IS_ENCRYPTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the connection is encrypted with secure communication enabled (SSL enabled): TRUE/ FALSE.</td>
</tr>
<tr>
<td>SSL_VERSION</td>
<td>VARCHAR(16)</td>
<td>Displays the SSL/TLS protocol version.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The result is empty if the connection does not use SSL (IS_ENCRYPTED) or if the SSL_VERSION contains the used SSL protocol version.</td>
</tr>
<tr>
<td>SSL_CIPHER</td>
<td>VARCHAR(64)</td>
<td>Displays the SSL cipher used for the connection.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The result is empty if SSL_CIPHER contains the used ciphersuite.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HAS_SSL_CLIENT_CERTIFICATE</td>
<td>VARCHAR(5)</td>
<td>Displays whether mutual authentication is enabled: TRUE/FALSE. TRUE indicates that the secure communication is enabled (SSL enabled) and the certificate of the client is validated. If not, FALSE is returned.</td>
</tr>
<tr>
<td>AUTHENTICATION_METHOD</td>
<td>VARCHAR(48)</td>
<td>Displays the name of the authentication method used to authenticate the connection.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time when the connection is closed for history connections.</td>
</tr>
<tr>
<td>PARENT_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the parent connection ID.</td>
</tr>
<tr>
<td>CLIENT_DISTRIBUTION_MODE</td>
<td>VARCHAR(128)</td>
<td>Displays the client distribution mode of the current connection (for example, [distribution] client_distribution_mode).</td>
</tr>
<tr>
<td>LOGICAL_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the logical connection ID in statement routing.</td>
</tr>
<tr>
<td>SOURCE_SITE_LOGICAL_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the logical connection ID of the origin site.</td>
</tr>
<tr>
<td>CURRENT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the current schema name.</td>
</tr>
<tr>
<td>CURRENT_THREAD_ID</td>
<td>BIGINT</td>
<td>Displays the current executing thread ID.</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>TINYINT</td>
<td>Displays the user-specified priority.</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the effective workload class.</td>
</tr>
<tr>
<td>CURRENT_COLLATION_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the name of the current collation.</td>
</tr>
<tr>
<td>CLOSE_REASON</td>
<td>VARCHAR(32)</td>
<td>Displays the reason for the connection close.</td>
</tr>
<tr>
<td>CLIENT_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the client interface name.</td>
</tr>
<tr>
<td>CLIENT_VERSION</td>
<td>NVARCHAR(256)</td>
<td>Displays the client version number.</td>
</tr>
<tr>
<td>CLIENT_APPLICATION</td>
<td>NVARCHAR(256)</td>
<td>Displays the client application name.</td>
</tr>
</tbody>
</table>
Related Information

M_CONNECTION_STATISTICS System View [page 1790]
CONNECT Statement (Session Management) [page 721]
HOST_CONNECTIONS View (Embedded Statistics Service)
Ports and Connections
Connection Types
SQL Connection Details
Pool of Remote Connections
Session-Specific Information for Connections
Edit an SQL Connection Configuration
Open a Support Connection
System Connection Properties

6.2.46 M_CONNECTION_STATISTICS System View

Provides detailed statistics on each connection between an application and database.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>SELECT_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of select statement executions.</td>
</tr>
<tr>
<td>SELECT_TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total execution time of select statement executions.</td>
</tr>
<tr>
<td>SELECT_AVG_EXECUTION_TIME</td>
<td>REAL</td>
<td>Displays the average execution time of select statement executions.</td>
</tr>
<tr>
<td>SELECT_MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum execution time of select statement executions.</td>
</tr>
<tr>
<td>SELECT_FOR_UPDATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of select for update executions.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SELECT_FOR_UPDATE_TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total execution time of select for update executions.</td>
</tr>
<tr>
<td>SELECT_FOR_UPDATE_AVG_EXECUTION_TIME</td>
<td>REAL</td>
<td>Displays the average execution time of select for update executions.</td>
</tr>
<tr>
<td>SELECT_FOR_UPDATE_MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum execution time of select for update execution.</td>
</tr>
<tr>
<td>SELECT_FOR_UPDATE_LOCK_WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of lock waits during select for update.</td>
</tr>
<tr>
<td>SELECT_FOR_UPDATE_TOTAL_LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total lock wait time during select for update.</td>
</tr>
<tr>
<td>SELECT_FOR_UPDATE_MAX_LOCK_WAIT_TIME</td>
<td>REAL</td>
<td>Displays the maximum lock wait time during select for update.</td>
</tr>
<tr>
<td>UPDATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of update statement and insert statement executions.</td>
</tr>
<tr>
<td>UPDATE_TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total execution time of update statement executions in microseconds.</td>
</tr>
<tr>
<td>UPDATE_AVG_EXECUTION_TIME</td>
<td>REAL</td>
<td>Displays the average execution time of update statement executions in micro-seconds.</td>
</tr>
<tr>
<td>UPDATE_MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum execution time of update statement executions in micro-seconds.</td>
</tr>
<tr>
<td>UPDATE_LOCK_WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of lock waits during update statement executions.</td>
</tr>
<tr>
<td>UPDATE_TOTAL_LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total lock wait execution time during update statement executions in microseconds.</td>
</tr>
<tr>
<td>UPDATE_AVG_LOCK_WAIT_TIME</td>
<td>REAL</td>
<td>Displays the average lock wait time during update statement executions in microseconds.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UPDATE_MAX_LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum lock wait time during update statement executions in microseconds.</td>
</tr>
<tr>
<td>READ_ONLY_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of read only transactions.</td>
</tr>
<tr>
<td>READ_ONLY_TRANSACTION_TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total execution time of read only transactions.</td>
</tr>
<tr>
<td>READ_ONLY_TRANSACTION_AVG_EXECUTION_TIME</td>
<td>REAL</td>
<td>Displays the average execution time of read only transactions in microseconds.</td>
</tr>
<tr>
<td>READ_ONLY_TRANSACTION_MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum execution time of read only transactions.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of update transactions.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total execution time of update transactions.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_AVG_EXECUTION_TIME</td>
<td>REAL</td>
<td>Displays the average execution time of update transactions.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum execution time of update transactions.</td>
</tr>
<tr>
<td>ROLLBACK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rolled back transactions.</td>
</tr>
<tr>
<td>ROLLBACK_TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total execution time of rollbacks.</td>
</tr>
<tr>
<td>ROLLBACK_AVG_EXECUTION_TIME</td>
<td>REAL</td>
<td>Displays the average execution time of rollbacks.</td>
</tr>
<tr>
<td>ROLLBACK_MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum execution time of rollbacks.</td>
</tr>
<tr>
<td>OTHERS_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of other statement executions including data definition statements and data control statements.</td>
</tr>
<tr>
<td>OTHERS_TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total execution time of other statements.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>OTHERS_AVG_EXECUTION_TIME</td>
<td>REAL</td>
<td>Displays the average execution time of other statements.</td>
</tr>
<tr>
<td>OTHERS_MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum execution time of other statements.</td>
</tr>
<tr>
<td>OTHERS_LOCK_WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total lock wait count of other statements.</td>
</tr>
<tr>
<td>OTHERS_TOTAL_LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total lock wait time of other statements.</td>
</tr>
<tr>
<td>OTHERS_AVG_LOCK_WAIT_TIME</td>
<td>REAL</td>
<td>Displays the average lock wait time of other statements.</td>
</tr>
<tr>
<td>OTHERS_MAX_LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum lock wait time of other statements.</td>
</tr>
<tr>
<td>LAST_EXECUTED_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last execution timestamp with this connection.</td>
</tr>
<tr>
<td>AVG_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the average memory size used during each execution in bytes.</td>
</tr>
<tr>
<td>MAX_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum memory size used during each execution in bytes.</td>
</tr>
<tr>
<td>MIN_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum memory size used during each execution in bytes.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the sum of the memory size used during each execution in bytes.</td>
</tr>
<tr>
<td>AVG_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of the statement preparation.</td>
</tr>
<tr>
<td>MAX_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of the statement preparation.</td>
</tr>
<tr>
<td>MIN_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of the statement preparation.</td>
</tr>
<tr>
<td>TOTAL_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of the statement preparation.</td>
</tr>
<tr>
<td>TOTAL_PREPARATION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total count of statement preparation.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EXECUTION_COUNT_BY_ROUTING</td>
<td>BIGINT</td>
<td>Displays the execution count by the client routed connection in statement routing.</td>
</tr>
<tr>
<td>COMMIT_MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of the commit execution.</td>
</tr>
<tr>
<td>COMMIT_AVG_EXECUTION_TIME</td>
<td>REAL</td>
<td>Displays the average time of the commit execution.</td>
</tr>
<tr>
<td>COMMIT_TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of the commit duration.</td>
</tr>
<tr>
<td>COMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the completed commit count.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time when the connection is closed for history connections.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_CONNECTIONS System View [page 1785]
- CONNECT Statement (Session Management) [page 721]
- HOST_CONNECTIONS View (Embedded Statistics Service)
- Ports and Connections
- Connection Types
- SQL Connection Details
- Pool of Remote Connections
- Session-Specific Information for Connections
- Edit an SQL Connection Configuration
- Open a Support Connection
- System Connection Properties
6.2.47 M_CONSISTENCY_CHECK_HISTORY System View

Provides table check run information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK_EXECUTION_ID</td>
<td>BIGINT</td>
<td>Displays the unique identifier for a run.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the connection ID of the last check execution.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the ID of the user performing the consistency check.</td>
</tr>
<tr>
<td>CHECK_PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the check procedure, for example CHECK_TABLE_CONSISTENCY.</td>
</tr>
<tr>
<td>CHECK_ACTION</td>
<td>VARCHAR(256)</td>
<td>Displays the single check action or a comma-separated list of check actions.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name. NULL is used if all schemas were checked.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>NVARCHAR(32)</td>
<td>Displays the type of the object.</td>
</tr>
<tr>
<td>CHECK_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of executions related to the aggregated entry.</td>
</tr>
<tr>
<td>FIRST_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the first start time of an aggregated entry.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For an aggregated entry without any errors, this value defines the interval in which all checks were error-free.</td>
</tr>
<tr>
<td>LAST_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last invocation start time of an aggregated entry.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For an aggregated entry without any errors, this value defines the interval in which all checks were error-free.</td>
</tr>
<tr>
<td>LAST_DURATION</td>
<td>BIGINT</td>
<td>Displays the total duration of the last check in milliseconds.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MIN_DURATION</td>
<td>BIGINT</td>
<td>Displays the minimum duration of the check runs in milliseconds.</td>
</tr>
<tr>
<td>MAX_DURATION</td>
<td>BIGINT</td>
<td>Displays the maximum duration of the check runs in milliseconds.</td>
</tr>
<tr>
<td>AVG_DURATION</td>
<td>BIGINT</td>
<td>Displays the average duration of the check runs milliseconds.</td>
</tr>
<tr>
<td>LAST_SCHEDULED_TABLE_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of tables scheduled for checking.</td>
</tr>
<tr>
<td>MIN_SCHEDULED_TABLE_COUNT</td>
<td>INTEGER</td>
<td>Displays the minimum number of scheduled table counts for the checks runs.</td>
</tr>
<tr>
<td>MAX_SCHEDULED_TABLE_COUNT</td>
<td>INTEGER</td>
<td>Displays the maximum number of scheduled table counts for the checks runs.</td>
</tr>
<tr>
<td>AVG_SCHEDULED_TABLE_COUNT</td>
<td>INTEGER</td>
<td>Displays the average number of scheduled table counts for the checks runs.</td>
</tr>
<tr>
<td>EXECUTED_TABLE_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of tables that have been checked.</td>
</tr>
<tr>
<td>ERROR_TABLE_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of tables with errors.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the error code returned by the check.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the error message returned by the check.</td>
</tr>
</tbody>
</table>

**Additional Information**

Further information about errors found during check runs can be found in the M_CONSISTENCY_CHECK_HISTORY_ERRORS system view.

Entries are aggregated, meaning that an existing entry is updated if the same checks were executed by the same user producing the same result.
Related Information

M_CONSISTENCY_CHECK HISTORY_ERRORS System View [page 1797]
HISTORY COLUMN Option (Time Travel) [page 875]
SET HISTORY SESSION Statement (Session Management) [page 1165]
Table Consistency Check
Table and Catalog Consistency Checks
Partitioning Consistency Check
Catalog Consistency Check
Configuration Parameters for the Table Consistency Check
SAP HANA History Tables

6.2.48 M_CONSISTENCY_CHECK_HISTORY_ERRORS System View

Lists the errors that were found within a specified check run.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECK_EXECUTION_ID</td>
<td>BIGINT</td>
<td>Displays the execution ID of the consistency check run that produced the error.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>NVARCHAR(32)</td>
<td>Displays the type of the object.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Contains 0 for an unpartitioned table.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the error code.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the error message.</td>
</tr>
<tr>
<td>SEVERITY</td>
<td>VARCHAR(8)</td>
<td>Displays the error severity: LOW, INFO, HIGH, or ERROR.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CHECK_ACTION</td>
<td>VARCHAR(64)</td>
<td>Displays the check action that produced the error. Contains NULL for errors not related to a specific check action.</td>
</tr>
<tr>
<td>AFFECTED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows affected by the error (optional). This column does not always contain a value, even in the case of an error. For some errors, this column may contain a NULL value.</td>
</tr>
<tr>
<td>DETAILS</td>
<td>NCLOB (JSON)</td>
<td>Displays the detailed information related to the error (optional). This column does not always contain a value, even in the case of an error. For some errors, this column may contain a NULL value.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_CONSISTENCY_CHECK_HISTORY System View [page 1795]
- SQL Error Codes [page 1212]
- HISTORY COLUMN Option (Time Travel) [page 875]
- SET HISTORY SESSION Statement (Session Management) [page 1165]
- Table Consistency Check
- Table and Catalog Consistency Checks
- Partitioning Consistency Check
- Catalog Consistency Check
- Configuration Parameters for the Table Consistency Check
- SAP HANA History Tables
- Break on Error
- Supported Error Codes
6.2.49 M_CONTAINER_DIRECTORY System View

Provides container directory statistics.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>CNT_CREATE</td>
<td>BIGINT</td>
<td>Displays the number of creates.</td>
</tr>
<tr>
<td>CNT_CREATE_ROLLBACK</td>
<td>BIGINT</td>
<td>Displays the number of rolled back creates.</td>
</tr>
<tr>
<td>CNT_REMOVE</td>
<td>BIGINT</td>
<td>Displays the number of removes.</td>
</tr>
<tr>
<td>CNT_REMOVE_FAIL</td>
<td>BIGINT</td>
<td>Displays the number of failed removes.</td>
</tr>
<tr>
<td>CNT_REMOVE_ROLLBACK</td>
<td>BIGINT</td>
<td>Displays the number of rolled back removes.</td>
</tr>
<tr>
<td>CNT_MOVE</td>
<td>BIGINT</td>
<td>Displays the number of moves.</td>
</tr>
<tr>
<td>CNT_GET_PHYSICALSIZE</td>
<td>BIGINT</td>
<td>Displays the number of getPhysicalSize.</td>
</tr>
<tr>
<td>CNT_GET</td>
<td>BIGINT</td>
<td>Displays the number of retrieved containers.</td>
</tr>
<tr>
<td>CNT_GET_FAIL</td>
<td>BIGINT</td>
<td>Displays the number of failed gets.</td>
</tr>
<tr>
<td>CNT_BEGIN</td>
<td>BIGINT</td>
<td>Displays the number of used iterators.</td>
</tr>
<tr>
<td>CNT_ITERATED</td>
<td>BIGINT</td>
<td>Displays the number of iterated containers.</td>
</tr>
<tr>
<td>CNT_CACHEHIT</td>
<td>BIGINT</td>
<td>Displays the number of cache hits.</td>
</tr>
<tr>
<td>CNT_CACHEMISS</td>
<td>BIGINT</td>
<td>Displays the number of cache misses.</td>
</tr>
<tr>
<td>CNT_CONTAINERS</td>
<td>BIGINT</td>
<td>Displays the number of existing containers.</td>
</tr>
</tbody>
</table>
6.2.50 M_CONTAINER_NAME_DIRECTORY System View

Provides ContainerNameDirectory statistics.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td></td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td></td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>CNT_CREATE</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the create counts.</td>
</tr>
<tr>
<td>CNT_CREATE_FAIL</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of failed creates.</td>
</tr>
<tr>
<td>CNT_INITIAL_CREATE</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of creates on the load.</td>
</tr>
<tr>
<td>CNT_INITIAL_SKIP</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of skips on the load.</td>
</tr>
<tr>
<td>CNT_REMOVE</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of removes.</td>
</tr>
<tr>
<td>CNT_REMOVE_FAIL</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of failed removes.</td>
</tr>
<tr>
<td>CNT_REMOVE_ALL</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of removeAll.</td>
</tr>
<tr>
<td>CNT_RENAME</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of renames.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>CNT_RENAME_FAIL</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of failed renames.</td>
</tr>
<tr>
<td>CNT_EXISTS</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of checked containers.</td>
</tr>
<tr>
<td>CNT_GET</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of retrieved containers.</td>
</tr>
<tr>
<td>CNT_GET_FAIL</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of failed gets.</td>
</tr>
<tr>
<td>CNT_BEGIN</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of used iterators.</td>
</tr>
<tr>
<td>CNT_ITERATE</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of iterated containers.</td>
</tr>
<tr>
<td>CNT_CONTAINERS</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of existing containers.</td>
</tr>
</tbody>
</table>

Related Information

M_CONTAINER_DIRECTORY System View [page 1799]
CONTAINS Predicate [page 64]
Monitoring Your Databases in the Database Directory
Logic Container

6.2.51 M_CONTEXT_MEMORY System View

Provides memory allocator statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence column ID.</td>
</tr>
<tr>
<td>STATISTICS_ID</td>
<td>BIGINT</td>
<td>Displays the statistics object unique ID.</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>VARCHAR(128)</td>
<td>Displays the allocator name.</td>
</tr>
<tr>
<td>DEPTH</td>
<td>BIGINT</td>
<td>Displays the depth.</td>
</tr>
<tr>
<td>INCLUSIVE_SIZE_IN_USE</td>
<td>BIGINT</td>
<td>Displays the current size of this allocator, including suballocators in bytes.</td>
</tr>
<tr>
<td>INCLUSIVE_COUNT_IN_USE</td>
<td>BIGINT</td>
<td>Displays the number of blocks currently in use, including suballocators.</td>
</tr>
<tr>
<td>INCLUSIVE_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total allocated size in this allocator and suballocators in bytes.</td>
</tr>
<tr>
<td>INCLUSIVE_DEALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total deallocated size in this allocator and suballocators in bytes.</td>
</tr>
<tr>
<td>INCLUSIVE_ALLOCATED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of allocations, including suballocators.</td>
</tr>
<tr>
<td>INCLUSIVE_DEALLOCATED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deallocations, including suballocators.</td>
</tr>
<tr>
<td>INCLUSIVE_MAX_SINGLE_ALLOCATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest block size, in bytes, ever allocated in this allocator and suballocators.</td>
</tr>
<tr>
<td>INCLUSIVE_PEAK_ALLOCATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest size of this allocator and suballocators (estimate) in bytes.</td>
</tr>
<tr>
<td>INCLUSIVE_LIMIT</td>
<td>BIGINT</td>
<td>Displays the maximum allowed memory size of the specified allocator and suballocators in bytes. The limit is not a hard limit and may therefore be violated slightly. It is not possible to set limits for each allocator individually.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INCLUSIVE_IN_USE_INTEGRAL</td>
<td>BIGINT</td>
<td>Displays the average allocated memory by this allocator and its suballocators, multiplied by the time since the start of measurement (sample based rough estimate). Deactivated by default, this value should only be activated upon request by SAP support. The UOM is 1 byte times 1 second.</td>
</tr>
<tr>
<td>EXCLUSIVE_SIZE_IN_USE</td>
<td>BIGINT</td>
<td>Displays the current size of this allocator in bytes.</td>
</tr>
<tr>
<td>EXCLUSIVE_COUNT_IN_USE</td>
<td>BIGINT</td>
<td>Displays the number of blocks currently in use.</td>
</tr>
<tr>
<td>EXCLUSIVE_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total allocated size in this allocator in bytes.</td>
</tr>
<tr>
<td>EXCLUSIVE_DEALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total deallocated size in this allocator in bytes.</td>
</tr>
<tr>
<td>EXCLUSIVE_ALLOCATED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of allocations.</td>
</tr>
<tr>
<td>EXCLUSIVE_DEALLOCATED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deallocations.</td>
</tr>
<tr>
<td>EXCLUSIVE_MAX_SINGLE_ALLOCATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest block size, in bytes, ever allocated in this allocator.</td>
</tr>
<tr>
<td>EXCLUSIVE_PEAK_ALLOCATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest size of this allocator (estimate) in bytes.</td>
</tr>
<tr>
<td>EXCLUSIVE_ALLOC_ERRORS</td>
<td>BIGINT</td>
<td>Displays the number of allocation errors.</td>
</tr>
<tr>
<td>EXCLUSIVE_IN_USE_INTEGRAL</td>
<td>BIGINT</td>
<td>Displays the average allocated memory by this allocator in bytes, multiplied by the time since the start of measurement (sample based rough estimate). Usually deactivate, this value should only be activated at the requested by SAP support. The UOM is 1 byte times 1 second.</td>
</tr>
<tr>
<td>MALLOC_PROXY_CACHE_MISSES</td>
<td>BIGINT</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>FLAGS</td>
<td>VARCHAR(64)</td>
<td>Displays the allocator flags.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VARCHAR(64)</td>
<td>Displays the name of the SAP HANA component of this allocator.</td>
</tr>
</tbody>
</table>
Additional Information

This view contains information about memory consumption grouped by connections and users. It does not contain information about all memory, only on memory that can be uniquely associated with either a connection, a statement, or a user.

To see information about allocated memory by component, use the M_HEAP MEMORY system view.

This view has a resettable counterpart; you can see the values since the last reset in the M_CONTEXT_MEMORY_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_CONTEXT_MEMORY_RESET;
```

Related Information

M_CONTEXT_MEMORY_RESET System View [page 1804]
M_HEAP MEMORY System View [page 1924]
M_MEMORY System View [page 2007]
M_CATALOG_MEMORY System View [page 1769]
Persistent Memory
Memory Management
Memory Sizing
Memory Analysis
Reset Peak Used Memory
Memory Allocator Statistics
View Memory Usage
Managing Memory by Object Usage

6.2.52 M_CONTEXT_MEMORY_RESET System View

Provides memory allocator statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_CONTEXT_MEMORY. Refer to M_CONTEXT_MEMORY for information about the structure and use of this view.

In addition to the members mentioned in M_CONTEXT_MEMORY, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_CONTEXT_MEMORY System View [page 1801]
6.2.53 M_CONVERTER_STATISTICS System View

Provides converter statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of converter.</td>
</tr>
<tr>
<td>MAX_LEVEL</td>
<td>BIGINT</td>
<td>Displays the maximum level. For example, the root page level.</td>
</tr>
<tr>
<td>MAX_PAGENUMBER</td>
<td>BIGINT</td>
<td>Displays the maximum page number in HEXID.</td>
</tr>
<tr>
<td>Allocated_PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of currently allocated pages.</td>
</tr>
<tr>
<td>Allocated_PAGE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size, in bytes, of the currently allocated pages.</td>
</tr>
<tr>
<td>MAX_ALLOCATED_PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum number of allocated pages.</td>
</tr>
<tr>
<td>MAX_ALLOCATED_PAGE_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size, in bytes, of the allocated pages.</td>
</tr>
<tr>
<td>AlLOCATE_PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of page allocations.</td>
</tr>
<tr>
<td>AlLOCATE_OR_GET_STATIC_PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of page allocations or retrievals during the static phase.</td>
</tr>
<tr>
<td>DEALLOCATE_PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of page deallocations.</td>
</tr>
<tr>
<td>ASSIGN_PHYSICAL_PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of physical page assignments.</td>
</tr>
<tr>
<td>UNASSIGN_PHYSICAL_PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of physical page unassignments.</td>
</tr>
</tbody>
</table>
### Column Name Table

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNASSIGN_PHYSICAL_PAGE_COUNT_DURING_DROP_SNAPSHOT</td>
<td>BIGINT</td>
<td>Displays the number of physical page unassignments during a drop snapshot.</td>
</tr>
<tr>
<td>CREATE_SNAPSHOT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of created snapshots.</td>
</tr>
<tr>
<td>DROP_SNAPSHOT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of dropped snapshots.</td>
</tr>
<tr>
<td>WRITE_CONVERTER_PAGE_LEVEL0_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of level 0 converter pages written to the disk.</td>
</tr>
<tr>
<td>WRITE_CONVERTER_PAGE_LEVEL1_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of level 1 converter pages written to the disk.</td>
</tr>
<tr>
<td>WRITE_CONVERTER_PAGE_LEVEL2_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of level 2 converter pages written to the disk.</td>
</tr>
<tr>
<td>WRITE_CONVERTER_PAGE_LEVEL3_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of level 3 converter pages written to the disk.</td>
</tr>
<tr>
<td>WRITE_CONVERTER_PAGE_LEVEL4_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of level 4 converter pages written to the disk.</td>
</tr>
</tbody>
</table>

### Additional Information

This view contains information about the converter, which administers logical page numbers and maps them to physical pages within the DataVolumes.

This view has a resettable counterpart; you can see the values since the last reset in the `M_CONVERTER_STATISTICS_RESET` system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_CONVERTER_STATISTICS_RESET;`

### Related Information

- `M_CONVERTER_STATISTICS_RESET` System View [page 1807]
- `M_PAGEACCESS_STATISTICS` System View [page 2037]
- `M_DATA_VOLUME_PAGE_STATISTICS` System View [page 1855]
- `CREATE STATISTICS` Statement (Data Definition) [page 815]
- `ALTER STATISTICS` Statement (Data Definition) [page 484]
- `REFRESH STATISTICS` Statement (Data Definition) [page 1082]
- `DROP STATISTICS` Statement (Data Definition) [page 974]
6.2.54 M_CONVERTER_STATISTICS_RESET System View

Provides converter statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_CONVERTER_STATISTICS. Refer to M_CONVERTER_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_CONVERTER_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_CONVERTER_STATISTICS System View [page 1805]

6.2.55 M_CS_ALL_COLUMNS System View

Provides runtime information for all columns in column tables, including internal column tables.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Returns the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For replicated tables, the part ID is 1 for the original table, and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID -1 indicates that the table schema is being modified.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>Displays the sum of the MEMORY_SIZE_IN_MAIN and MEMORY_SIZE_IN_DELTA columns in bytes.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_MAIN</td>
<td>BIGINT</td>
<td>Displays the current memory consumption in main in bytes. Returns 0 if the memory is not loaded in main.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_DELTA</td>
<td>BIGINT</td>
<td>Displays the current memory consumption in delta in bytes. Returns 0 if the memory is not loaded in delta.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_PAGE_LOADABLE_MAIN</td>
<td>BIGINT</td>
<td>Displays the total paged memory size of the column in bytes.</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>For columns stored in persistent memory, displays the total memory size in bytes.</td>
</tr>
<tr>
<td>UNCOMPRESSED_SIZE</td>
<td>BIGINT</td>
<td>Displays the estimated, uncompressed column size in bytes.</td>
</tr>
<tr>
<td>COMPRESSION_RATIO_IN_PERCENTAGE</td>
<td>DOUBLE</td>
<td>Displays the compression ratio percentage. This value is determined by dividing the MEMORY_SIZE_IN_TOTAL by the UNCOMPRESSED_SIZE and multiplying the result by 100.</td>
</tr>
<tr>
<td>COUNT</td>
<td>BIGINT</td>
<td>Displays the record count. Returns 0 if the count is not loaded.</td>
</tr>
<tr>
<td>DISTINCT_COUNT</td>
<td>BIGINT</td>
<td>Displays the distinct count of values. Returns 0 if the count is not loaded.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COMPRESSION_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of column compression. This value can be: SPARSE, PREFIXED, CLUSTERED, INDIRECT, RLE, or DEFAULT (if the column is only dictionary coded). The columns in the M_CS_COLUMNS system view show the runtime value, which can be changed during the runtime.</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of inverted index:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>FULL</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displays that a FULL inverted index is defined on the column. A FULL index is a data structure that assigns a position to each value in the column. FULL indexes are the conventional form of index.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BLOCK</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displays that a block index is defined on the column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A BLOCK index is a data structure where each value in the column is assigned to one of a list of fixed-size blocks for the column. BLOCK indexes are typically much smaller than FULL indexes. Reverse look-ups on BLOCK indexes are effectively a combination of reading the index (fast) and scanning the read blocks (possibly slow, depending on the compression of the column). BLOCK indexes can be used with compressed columns.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>MINMAX</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displays that a min-max index is defined for the column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NONE</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displays that no index is defined for the column.</td>
</tr>
<tr>
<td>INDEX_LOADED</td>
<td>VARCHAR(16)</td>
<td>Displays the load status of the inverted index: NOT APPLICABLE, UNLOADED, or LOADED.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IMPLEMENTATION_FLAGS</td>
<td>BIGINT</td>
<td>Displays the column internal implementation specification summary.</td>
</tr>
<tr>
<td>LAST_ACCESS_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time that the column was read or there was an INSERT to, UPDATE of, or DELETE from the column. This value is undefined for unloaded columns.</td>
</tr>
<tr>
<td>LOADED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the flag to indicate that the column is loaded into memory (including DRAM or persistent memory): TRUE/FALSE.</td>
</tr>
<tr>
<td>LOADED_FROM_PERSISTENT_MEMORY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column was loaded from persistent memory: TRUE/FALSE.</td>
</tr>
<tr>
<td>STORED_IN_PERSISTENT_MEMORY</td>
<td>VARCHAR(5)</td>
<td>Displays whether there is an associated persistent memory block: TRUE/FALSE.</td>
</tr>
<tr>
<td>LOAD_UNIT</td>
<td>VARCHAR(7)</td>
<td>Displays the load unit for the columns: PAGE, COLUMN, and UNKNOWN.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>MAIN_MEMORY_SIZE_IN_DATA</td>
<td>BIGINT</td>
<td>Displays the current memory consumption in the data in bytes.</td>
</tr>
<tr>
<td>MAIN_MEMORY_SIZE_IN_DICT</td>
<td>BIGINT</td>
<td>Displays the current memory consumption in the dictionary in bytes.</td>
</tr>
<tr>
<td>MAIN_MEMORY_SIZE_IN_INDEX</td>
<td>BIGINT</td>
<td>Displays the current memory consumption in the index in bytes.</td>
</tr>
<tr>
<td>MAIN_MEMORY_SIZE_IN_MISC</td>
<td>BIGINT</td>
<td>Displays the miscellaneous current memory consumption in bytes.</td>
</tr>
<tr>
<td>DELTA_MEMORY_SIZE_IN_DATA</td>
<td>BIGINT</td>
<td>Displays the current memory consumption in the data in bytes.</td>
</tr>
<tr>
<td>DELTA_MEMORY_SIZE_IN_DICT</td>
<td>BIGINT</td>
<td>Displays the current memory consumption in the dictionary in bytes.</td>
</tr>
<tr>
<td>DELTA_MEMORY_SIZE_IN_INDEX</td>
<td>BIGINT</td>
<td>Displays the current memory consumption in the index in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DELTA_MEMORY_SIZE_IN_MISC</td>
<td>BIGINT</td>
<td>Displays the miscellaneous current memory consumption in bytes.</td>
</tr>
<tr>
<td>MAIN_PAGE_LOADABLE_MEMORY_SIZE_IN_DATA</td>
<td>BIGINT</td>
<td>Displays the total paged memory size of the column index vector in bytes.</td>
</tr>
<tr>
<td>MAIN_PAGE_LOADABLE_MEMORY_SIZE_IN_DICT</td>
<td>BIGINT</td>
<td>Displays the total paged memory of the column dictionary in bytes.</td>
</tr>
<tr>
<td>MAIN_PAGE_LOADABLE_MEMORY_SIZE_IN_INDEX</td>
<td>BIGINT</td>
<td>Displays the total paged memory of the column’s inverted index in bytes.</td>
</tr>
<tr>
<td>MAIN_PAGE_LOADABLE_MEMORY_SIZE_IN_MISC</td>
<td>BIGINT</td>
<td>Displays the total paged memory size of other column structures in bytes.</td>
</tr>
<tr>
<td>MAIN_PERSISTENT_MEMORY_SIZE_IN_DATA</td>
<td>BIGINT</td>
<td>For columns stored in persistent memory, displays the current memory consump-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tion in the data in bytes.</td>
</tr>
<tr>
<td>MAIN_PERSISTENT_MEMORY_SIZE_IN_DICT</td>
<td>BIGINT</td>
<td>For columns stored in persistent memory, displays the current memory consump-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tion in the dictionary in bytes.</td>
</tr>
<tr>
<td>MAIN_PERSISTENT_MEMORY_SIZE_IN_INDEX</td>
<td>BIGINT</td>
<td>For columns stored in persistent memory, displays the current memory consump-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tion in the index in bytes.</td>
</tr>
<tr>
<td>MAIN_PERSISTENT_MEMORY_SIZE_IN_MISC</td>
<td>BIGINT</td>
<td>For columns stored in persistent memory, displays the current memory consump-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tion not already accounted for in the other memory size columns in bytes.</td>
</tr>
<tr>
<td>MAIN_PERSISTENT_MEMORY_SIZE_UNUSED</td>
<td>BIGINT</td>
<td>Displays the amount of persistent memory, in bytes, that is not directly</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mapped to a corresponding data structure during load, but was only read to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>construct the data structure in DRAM. This column also includes persistent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>memory block overhead if the requested persistent memory allocation is not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a multiple of a block size.</td>
</tr>
<tr>
<td>INDEX_CREATE_DURATION</td>
<td>BIGINT</td>
<td>Displays the time needed to create an inverted index.</td>
</tr>
<tr>
<td>DELTA_IMPLEMENTATION_FLAGS</td>
<td>BIGINT</td>
<td>Displays the internal column implementation specification summary.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| INTERNAL_ATTRIBUTE_TYPE     | VARCHAR(32)  | Displays the attribute type for internal columns. Returns NULL for non-inter-
|                             |              | nal columns.                                                                 |
| LAST_LOAD_TIME              | TIMESTAMP    | Displays the last time that the column was loaded. This value is undefined for
|                             |              | unloaded columns.                                                           |
| PERSISTENT_MEMORY           | VARCHAR(5)   | Displays the persistent memory preference inferred from the user-specified
|                             |              | preferences.                                                                |
| NUMA_NODE_INDEX             | SMALLINT     | Displays the NUMA node index where allocations are performed and columns are
|                             |              | loaded.                                                                     |
| PERSISTENT_MEMORY_FILE_NAME | VARCHAR(256) | Displays the name of the persistent memory file associated with the col-
|                             |              | umn. If there is no associated persistent memory file, then NULL is re-
|                             |              | turned.                                                                     |

**Additional Information**

The information returned in the view is valid only for loaded columns (LOADED = TRUE).

**Related Information**

- M_CS_ALL_COLUMN_STATISTICS System View [page 1813]
- CS_ALL_COLUMNS System View [page 1525]
- CS_CONCAT_COLUMNS System View [page 1527]
- CS_VIEW_COLUMNS System View [page 1535]
- RENAME COLUMN Statement (Data Definition) [page 1088]
- Memory Management in the Column Store
- Data Compression in the Column Store
- Viewing Load Unit Information for Column Store Tables in SAP HANA NSE
- Understanding Load Unit Behavior in SAP HANA NSE Column Store Tables
- Monitoring View Extensions for Column Store Paged Data Size
## 6.2.56 M_CS_ALL_COLUMN_STATISTICS System View

Provides information on how many scans and index searches were performed on any specified columns.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Returns the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is currently in progress.</td>
</tr>
<tr>
<td>SCANNED_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of scanned rows on the main part of a specified column. The value is reset when the column is unloaded or merged.</td>
</tr>
<tr>
<td>INDEX_LOOKUP_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of index lookups performed on the main part of a specified column. The value is reset when the column is unloaded or merged.</td>
</tr>
</tbody>
</table>

### Related Information

- [M_CS_ALL_COLUMNS System View](#) [page 1807]
- [CS_ALL_COLUMNS System View](#) [page 1525]
- [CS_CONCAT_COLUMNS System View](#) [page 1527]
- [CS_VIEW_COLUMNS System View](#) [page 1535]
- [RENAME COLUMN Statement (Data Definition)](#) [page 1088]
- Memory Management in the Column Store
6.2.57 M_CS_COLUMNS System View

Provides runtime information about columns in column tables.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Returns the table partition ID:</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>Displays the sum, in bytes, of the memory used.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_MAIN</td>
<td>BIGINT</td>
<td>Displays the current memory consumption, in bytes, in main. Returns 0 if the memory is not loaded in main.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_DELTA</td>
<td>BIGINT</td>
<td>Displays the current memory consumption, in bytes, in delta. Returns 0 if the memory is not loaded in delta.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_PAGE_LOADABLE_MAIN</td>
<td>BIGINT</td>
<td>Displays the total paged memory size of the column in bytes.</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>For columns stored in persistent memory, displays the total memory size in bytes.</td>
</tr>
<tr>
<td>UNCOMPRESSED_SIZE</td>
<td>BIGINT</td>
<td>Displays the estimated column size, in bytes, if the column is not compressed.</td>
</tr>
<tr>
<td>COMPRESSION_RATIO_IN_PERCENTAGE</td>
<td>DOUBLE</td>
<td>Displays the compression ratio percentage. This value is determined by dividing the MEMORY_SIZE_IN_TOTAL by the UNCOMPRESSED_SIZE and multiplying the result by 100.</td>
</tr>
<tr>
<td>COUNT</td>
<td>BIGINT</td>
<td>Displays the record count. This value is -1 if the count is not loaded.</td>
</tr>
<tr>
<td>DISTINCT_COUNT</td>
<td>BIGINT</td>
<td>Displays a distinct count of values, generally. This value is 0 if the count is not loaded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The values in this column are not equal to actual values that might be returned by a SELECT DISTINCT query. Do not use these values in a production system if precision is required.</td>
</tr>
<tr>
<td>COMPRESSION_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of column compression:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SPARSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PREFIXED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CLUSTERED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• INDIRECT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DEFAULT (if the column is only dictionary coded)</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of inverted index:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>FULL</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displays that a conventional index is defined on the column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>BLOCK</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displays that a block index is defined on the column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>MINMAX</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displays that a min-max index is defined on the column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NONE</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displays that no index is defined on the column.</td>
</tr>
<tr>
<td>INDEX_LOADED</td>
<td>VARCHAR(16)</td>
<td>Displays the load status of the inverted index: NOT APPLICABLE, UNLOADED, or LOADED.</td>
</tr>
<tr>
<td>IMPLEMENTATION_FLAGS</td>
<td>BIGINT</td>
<td>Displays the internal implementation specification summary of the column.</td>
</tr>
<tr>
<td>LAST_ACCESS_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time that the column was read or there was an INSERT to, UPDATE of, or DELETE from the column. This value is undefined for unloaded columns.</td>
</tr>
<tr>
<td>LOADED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the column is loaded into the memory: TRUE/FALSE.</td>
</tr>
<tr>
<td>LOADED_FROM_PERSISTENT_MEMORY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column was loaded from persistent memory: TRUE/ FALSE.</td>
</tr>
<tr>
<td>STORED_IN_PERSISTENT_MEMORY</td>
<td>VARCHAR(5)</td>
<td>Displays whether there is an associated persistent memory block: TRUE/ FALSE.</td>
</tr>
<tr>
<td>LOAD_UNIT</td>
<td>VARCHAR(7)</td>
<td>Displays the load unit for the column: PAGE, COLUMN, and UNKNOWN.</td>
</tr>
<tr>
<td>LAST_LOAD_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time that the column was loaded. This is undefined for unloaded columns.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY</td>
<td>VARCHAR(5)</td>
<td>Displays the persistent memory preference inferred from the user-specified preferences in a bottom-up manner starting from this object: TRUE/FALSE.</td>
</tr>
<tr>
<td>NUMA_NODE_INDEX</td>
<td>SMALLINT</td>
<td>Displays the NUMA node index where allocations are performed and columns are loaded.</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY_FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the name of the persistent memory file associated with the column. If there is no associated persistent memory file, then NULL is returned.</td>
</tr>
</tbody>
</table>

**Additional Information**

The information returned in the view is valid only for loaded columns (LOADED = TRUE).

**Related Information**

- M_CS_ALL_COLUMNS System View [page 1807]
- CS_ALL_COLUMNS System View [page 1525]
- CS CONCAT_COLUMNS System View [page 1527]
- CS VIEW_COLUMNS System View [page 1535]
- RENAME COLUMN Statement (Data Definition) [page 1088]
- Memory Management in the Column Store
- Data Compression in the Column Store
- Viewing Load Unit Information for Column Store Tables in SAP HANA NSE
- Understanding Load Unit Behavior in SAP HANA NSE Column Store Tables
- Monitoring View Extensions for Column Store Paged Data Size
6.2.58  M_CS_COLUMNS_PERSISTENCE System View

Provides column persistence information for column tables.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Returns the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For replicated tables, the part ID is 1 for the original table and sub-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is currently in progress.</td>
</tr>
<tr>
<td>COLUMN_ID</td>
<td>INTEGER</td>
<td>Displays the numeric column ID.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>PERSISTENCE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of column persistence: SINGLE, PAGED, VIRTUAL_FILE, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VIRTUAL_PAGED.</td>
</tr>
<tr>
<td>MAIN_PHYSICAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the storage size, in bytes, used by the column.</td>
</tr>
<tr>
<td>MAIN_PHYSICAL_SIZE_IN_PAGE_LOADABLE</td>
<td>BIGINT</td>
<td>Displays the total on-disk size stored in a page-loadable format for the table in bytes.</td>
</tr>
</tbody>
</table>
## M_CS_INDEXES System View

Provides information for column store indexes.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the index name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the part ID is 1 for the original table and subsequent part IDs are assigned</td>
</tr>
<tr>
<td></td>
<td></td>
<td>to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in progress.</td>
</tr>
<tr>
<td>HASH_COLLISION_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of hash collisions aggregated over all values.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CONCAT_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the concat column in the case of multi-column indexes. This is empty in the case of a single-column index.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>Displays the total size of the concat plus the sum of the inverted index sizes of the single index columns in bytes.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_CONCAT</td>
<td>BIGINT</td>
<td>Displays the total size of the concat in bytes.</td>
</tr>
<tr>
<td>MOST_SELECTIVE_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the most selective index column of a composite index. The most selective column is the one with lowest cost value.</td>
</tr>
<tr>
<td>INVERTED_INDIVIDUAL_COST</td>
<td>BIGINT</td>
<td>Displays the cost value for INVERTED INDIVIDUAL index type. This cost value is an indicator for the possible performance impact when switching between different index-types. A high cost value indicates a possible high overhead when using INVERTED INDIVIDUAL index-type compared to INVERTED VALUE index-type. A cost value of 1 indicates that there is no overhead. For non composite or non-unique indexes the value is -1, which indicates that no cost value is available.</td>
</tr>
</tbody>
</table>

**Related Information**

INDEXES System View [page 1585]
CREATE INDEX Statement (Data Definition) [page 762]
ALTER INDEX Statement (Data Definition) [page 452]
DROP INDEX Statement (Data Definition) [page 958]
Changing the Load Units for Indexes Using ALTER INDEX
## 6.2.60 M_CS_LOADS System View

Provides a history of column loads.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>LOAD_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of the load event.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name. This is left empty if the whole table is loaded.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is in progress.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>IS_HISTORY</td>
<td>VARCHAR(5)</td>
<td>Displays the flag to indicate whether the history part is loaded (this is only relevant for history tables). This is left empty if the whole table is loaded.</td>
</tr>
<tr>
<td>LOAD_DURATION</td>
<td>BIGINT</td>
<td>Displays the load duration, in milliseconds.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the error code.</td>
</tr>
</tbody>
</table>
### Related Information

- LOAD Statement (Data Manipulation) [page 1054]
- Viewing Load Unit Information for Column Store Tables in SAP HANA NSE
- Understanding Load Unit Behavior in SAP HANA NSE Column Store Tables
- Load or Unload Tables

### 6.2.61 M_CS_LOB_SPACE_RECLAIMS System View

Provides information regarding executed LOB garbage collection runs.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the name of the host.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td></td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td></td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>COLLECTION_SCOPE</td>
<td>VARCHAR(8)</td>
<td></td>
<td>Displays the scope of the garbage collection run on either VOLUME, TABLES, or TABLE.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Displays the schema name of the scanned table if the scope is TABLE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Displays the name of the scanned table if the scope is TABLE.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td></td>
<td>Displays the start TIMESTAMP of the garbage collection run.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td></td>
<td>Displays the end TIMESTAMP of the garbage collection run.</td>
</tr>
<tr>
<td>SCANNED_COLUMNS_COUNT</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of overall scanned columns.</td>
</tr>
<tr>
<td>REMOVED_FILE_LOBS_COUNT</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of removed file LOBs.</td>
</tr>
<tr>
<td>REMOVED_PACKED_LOBS_COUNT</td>
<td>BIGINT</td>
<td>Counter</td>
<td>Displays the number of removed packed LOBs.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td></td>
<td>Displays the first error code if the execution fails. More details are available in the ERROR_MESSAGE field.</td>
</tr>
</tbody>
</table>

**Related Information**

- ALTER SYSTEM RECLAIM LOB SPACE Statement (System Management) [page 553]
- HOST_CS_LOB_SPACE_RECLAIMS View (Embedded Statistics Service)
- M_GARBAGE_COLLECTION_STATISTICS System View [page 1920]
- Reclaim Space
- Hybrid LOBs (Large Objects)
### 6.2.62 M_CS_LOG_REPLAY_QUEUE_STATISTICS System View

Provides information about column store log replay queue statistics.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the name of the host.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>LOG_REPLAY_QUEUE_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the recovery queue.</td>
</tr>
<tr>
<td>TOTAL_CS_LOG_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of redo records that were processed in this queue.</td>
</tr>
<tr>
<td>TOTAL_DML_LOG_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of column store DML redo records that were processed in this queue.</td>
</tr>
<tr>
<td>TOTAL_DDL_LOG RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of column store DDL redo records that were processed in this queue.</td>
</tr>
<tr>
<td>LAST_CS_LOG_RECORD_TYPE</td>
<td>VARCHAR(40)</td>
<td>Displays the last column store log record type in this queue.</td>
</tr>
<tr>
<td>LAST_LOG_REPLAY_POSITION</td>
<td>BIGINT</td>
<td>Displays the last log replay position that was processed in this queue.</td>
</tr>
<tr>
<td>LAST_SAVEPOINT_POSITION</td>
<td>BIGINT</td>
<td>Displays the last savepoint position that was passed in this queue.</td>
</tr>
<tr>
<td>TOTAL_SAVEPOINT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of savepoints that were passed in this queue.</td>
</tr>
<tr>
<td>ACTIVE_TABLE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of tables that are currently handled in this queue.</td>
</tr>
<tr>
<td>ACTIVE_DELTA MERGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of delta merges that are currently ongoing for tables that are handled in this queue.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TOTAL_DELTA_MERGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of delta merges that were started on this queue.</td>
</tr>
<tr>
<td>TOTAL_SUCCESSFUL_DELTA_MERGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of successful merges.</td>
</tr>
<tr>
<td>TOTAL_DELTA_FUSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of delta fusions that were made in this queue.</td>
</tr>
<tr>
<td>ACTIVE_OPTIMIZE_COMPRESSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of optimize compressions that are currently ongoing for tables that are handled in this queue.</td>
</tr>
<tr>
<td>TOTAL_OPTIMIZE_COMPRESSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of successful optimize compressions.</td>
</tr>
<tr>
<td>TOTAL_SUCCESSFUL_OPTIMIZE_COMPRESSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of successful optimize compressions.</td>
</tr>
<tr>
<td>TOTAL_DML_CONTEXT_CREATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of expensive creations of DML contexts.</td>
</tr>
</tbody>
</table>

**Related Information**

- **M_CS_LOG_REPLAY_QUEUE_STATISTICS_RESET System View [page 1825]**
- HOST_SERVICE_REPLICATION View (Embedded Statistics Service)
- Logging
- Replaying a Workload
- Replay a Preprocessed Workload
- Replaying Configuration Settings
- Generate a Replay-Replay Comparison Report
- Capturing and Replaying Workloads

**6.2.63 M_CS_LOG_REPLAY_QUEUE_STATISTICS_RESET System View**

Provides information about column store log replay queue statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_CS_LOG_REPLAY_QUEUE_STATISTICS_RESET. Refer to M_CS_LOG_REPLAY_QUEUE_STATISTICS_RESET for information about the structure and use of this view.

In addition to the members mentioned in M_CS_LOG_REPLAY_QUEUE_STATISTICS_RESET, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.
Related Information

M_CS_LOG_REPLAY_QUEUE_STATISTICS System View [page 1824]

6.2.64  M_CS_MVCC System View

Provides column store MVCC information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port number.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
</tbody>
</table>
| PART_ID        | INTEGER      | Displays the partition ID. Returns the following:
<p>|                |              | • For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1. |
|                |              | • In the case of replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables. |
|                |              | • The part ID is 0 for tables that are not partitioned. |
|                |              | • A part ID value of -1 indicates that a modification of the table schema is in progress. |
| IS_HISTORY     | VARCHAR(5)   | Indicates if the partition is a history partition: TRUE/FALSE. |
| LOADED         | VARCHAR(5)   | Indicates if the MVCC data is loaded into the memory. |
| FRAGMENT_TYPE  | VARCHAR(7)   | Displays the fragment type: DELTA1, DELTA2, or MAIN. |</p>
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of visible records in the table.</td>
</tr>
<tr>
<td>RAW_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of physical records in the table.</td>
</tr>
<tr>
<td>ROWSTATE_BLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active rowstate blocks.</td>
</tr>
<tr>
<td>CTS_BLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of blocks used to store creation timestamps (CTS).</td>
</tr>
<tr>
<td>DTS_BLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of blocks used to store deletion timestamps (DTS).</td>
</tr>
<tr>
<td>ROWSTATE_STUB_BLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of stub blocks, both visible and invisible, present in the MVCC layer associated with the fragment.</td>
</tr>
<tr>
<td>VISIBLE_ROWSTATE_STUB_BLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of visible stub blocks present in the MVCC layer associated with the fragment.</td>
</tr>
<tr>
<td>CTS_STUB_BLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active CTS stub blocks.</td>
</tr>
<tr>
<td>DTS_STUB_BLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active DTS stub blocks.</td>
</tr>
<tr>
<td>FREE_ROWSTATE_BLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of free rowstate blocks available for successive operations.</td>
</tr>
<tr>
<td>FREE_TS_BLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of free CTS/DTS blocks.</td>
</tr>
<tr>
<td>FREE_TS_STUB_BLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of free CTS/DTS stub blocks.</td>
</tr>
<tr>
<td>PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of pages in the MVCC page chain.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>If MVCC data is loaded, then it displays the total memory size of MVCC structures in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PAGE_CHAIN_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>If MVCC data is loaded, then it displays the memory size of the pinned MVCC page chain in bytes.</td>
</tr>
<tr>
<td>PAGE_CHAIN_ACTIVE_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size of the active part of the MVCC page chain in bytes.</td>
</tr>
<tr>
<td>ROWSTATE_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>If MVCC data is loaded, then it displays the memory size of transient rowstate structures in bytes.</td>
</tr>
<tr>
<td>CTS_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the amount of memory used to store CTS blocks in bytes.</td>
</tr>
<tr>
<td>DTS_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the amount of memory used to store DTS blocks in bytes.</td>
</tr>
<tr>
<td>MISC_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>If MVCC data is loaded, then it displays the memory size of miscellaneous MVCC related structures in bytes.</td>
</tr>
<tr>
<td>SPARSE_DTS_BLOCK_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of DTS blocks that are sparse timestamp blocks.</td>
</tr>
<tr>
<td>SPARSE_DTS_MEMORY_SIZE</td>
<td>INTEGER</td>
<td>Displays the memory size of the sparse block in bytes.</td>
</tr>
<tr>
<td>FREE_SPARSE_TS_BLOCK_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of available sparse timestamp blocks for successive operations.</td>
</tr>
</tbody>
</table>

**Related Information**

- CURRENT_MVCC_SNAPSHOT_TIMESTAMP Function (Datetime) [page 168]
- M_MVCC_SNAPSHOTS System View [page 2029]
- M_MVCC_OVERVIEW System View [page 2027]
- HOST_MVCC_OVERVIEW View (Embedded Statistics Service)
- M_MVCC_TABLES System View [page 2030]
- Monitoring Multi-Host Health
6.2.65 M_CS_NSE_ADVISOR System View

Provides an SQL interface of the NSE Recommendation Engine and advice on changing the load unit (page/column) for certain tables in the SAP HANA system, balancing out performance and memory – thus achieving a low total cost of ownership.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name. When the recommendation level is not column, this field is not applicable and the value NULL is used.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the table partition ID. When the recommendation level is not Partition, this field is not applicable, and the value is 0.</td>
</tr>
<tr>
<td>LOAD_UNIT</td>
<td>VARCHAR(7)</td>
<td>Displays the recommended load unit for this object: COLUMN/PAGE.</td>
</tr>
<tr>
<td>GRANULARITY</td>
<td>VARCHAR(10)</td>
<td>Displays the object level at which the recommendation for this table is given: TABLE, PARTITION, or COLUMN.</td>
</tr>
<tr>
<td>CONFIDENCE</td>
<td>INTEGER</td>
<td>Displays the confidence value of the NSE Advisor. The value here can be used to objectively compare recommendations of a single NSE Advisor run.</td>
</tr>
</tbody>
</table>

i Note

Confidence can only be used to compare advices for same run of NSE Advisor.
### related information

Viewing Load Unit Information for Column Store Tables in SAP HANA NSE
Understanding the SAP HANA NSE Advisor

### 6.2.66 M_CS_NSE_ADVISOR_DETAILS System View

Provides details of all objects included in the workload evaluated by the NSE advisor.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays object ID of the table.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the table partition ID. If the recommendation is not on the partition level, this field is not applicable, and the value 0 is used.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name. If the recommendation is not on the column level, this field is not applicable, and the value NULL is used.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COLUMN_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the column.</td>
</tr>
<tr>
<td>GRANULARITY</td>
<td>VARCHAR(10)</td>
<td>Displays the object level at which the recommendation for this table is given: TABLE, PARTITION, or COLUMN.</td>
</tr>
<tr>
<td>SCAN_COUNT</td>
<td>BIGINT</td>
<td>Displays count of scans.</td>
</tr>
<tr>
<td>MATERIALIZATION_COUNT</td>
<td>BIGINT</td>
<td>Displays count of materialization.</td>
</tr>
<tr>
<td>CONFIDENCE</td>
<td>INTEGER</td>
<td>Displays the confidence value of the NSE Advisor. The value here can be used to objectively compare advice of a single NSE Advisor run. The object is not advised if the value is -1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>i Note</strong> Confidence can only be used to compare advice for same run of NSE Advisor.</td>
</tr>
<tr>
<td>CURRENT_LOAD_UNIT</td>
<td>VARCHAR(8)</td>
<td>Displays the current load unit for this object: COLUMN/PAGE.</td>
</tr>
<tr>
<td>TARGET_LOAD_UNIT</td>
<td>VARCHAR(8)</td>
<td>Displays the recommended load unit for this object: COLUMN/PAGE.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_MAIN</td>
<td>BIGINT</td>
<td>Displays the memory consumption size, in bytes, per granularity of the object in main memory. If not LOADED or gathered, the value is 0 is used.</td>
</tr>
<tr>
<td>MAIN_PHYSICAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the storage size, in bytes, per granularity of the object. If not gathered, the value 0 is used.</td>
</tr>
<tr>
<td>SKIP_REASON</td>
<td>VARCHAR(512)</td>
<td>Displays reason why this object is skipped and not advised. See next section for all possible values for this column.</td>
</tr>
</tbody>
</table>

**SKIP_REASON Column**

**i Note**
The output name and description are displayed as one string.
<table>
<thead>
<tr>
<th>Output Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SYSTEM TABLE]</td>
<td>The table is not advised because it is a system table or a table in schema '_SYS' or 'SYS'.</td>
</tr>
<tr>
<td>[TABLE NOT SUPPORTED IN NSE]</td>
<td>The table is not advised because it does not support NSE, for example a TEMPORARY table or a NO LOGGING table.</td>
</tr>
<tr>
<td>[EXCLUDED BY FILTER RULE]</td>
<td>The object is not advised because it is excluded by filter rules.</td>
</tr>
<tr>
<td>[SMALLER THAN MIN_OBJECT_SIZE]</td>
<td>The object is not advised because it is smaller than the MIN_OBJECT_SIZE threshold.</td>
</tr>
<tr>
<td>[NO ACCESS COUNT]</td>
<td>The column or partition has no access counters, therefore the column or partition level recommendation is not generated.</td>
</tr>
<tr>
<td>[SCAN DENSITY]</td>
<td>The object is not advised because it is not considered hot nor cold data based on its scan density.</td>
</tr>
<tr>
<td>[CONFLICT WITH COLUMN OR PARTITION ADVICE]</td>
<td>The table is not advised because the calculated recommendation is in conflict with at least one column or partition level recommendation of the same table.</td>
</tr>
<tr>
<td>[COVERED BY TABLE LEVEL ADVICE]</td>
<td>The column or partition level recommendation is not advised because the calculated recommendation is the same as on table level.</td>
</tr>
<tr>
<td>[NO CHANGE NEEDED FOR LOAD UNIT]</td>
<td>The object is not advised because the current load unit configuration already matches the calculated recommendation.</td>
</tr>
<tr>
<td>[NOT HANDLED BY THIS INDEXSERVER]</td>
<td>The table is partitioned across different multiple nodes in a HANA scale-out system. Only one node will calculate recommendations for the table.</td>
</tr>
<tr>
<td>[INTERNAL ERROR]</td>
<td>The table is not advised because an internal error occurs when getting the meta data on the table.</td>
</tr>
</tbody>
</table>

**Related Information**

Understanding the SAP HANA NSE Advisor

**6.2.67 M_CS_NSE_ADVISOR_STATISTICS System View**

Provides statistics for NSE Advisor.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REQUEST_ID</td>
<td>INTEGER</td>
<td>Displays an increasing index of requests. Start from 1 and will loop back from 1 when overflow.</td>
</tr>
<tr>
<td>ALGORITHM</td>
<td>NVARCHAR(6)</td>
<td>Displays algorithm name.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays start time of run of NSE Advisor.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays end time of run of NSE advisor. If this run has not finished, it will be NULL.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays connection ID that initializes run of NSE advisor.</td>
</tr>
<tr>
<td>CURRENT_PROGRESS</td>
<td>INTEGER</td>
<td>Displays estimated progress of this run.</td>
</tr>
<tr>
<td>MAX_PROGRESS</td>
<td>INTEGER</td>
<td>Displays the maximum operation progress.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the error code.</td>
</tr>
<tr>
<td>RECOMMENDATION_COUNT</td>
<td>INTEGER</td>
<td>Displays how many recommendations are produced in the run.</td>
</tr>
<tr>
<td>HOT_OBJECT_THRESHOLD</td>
<td>TINYINT</td>
<td>Displays hot_object_threshold used in the run.</td>
</tr>
<tr>
<td>COLD_OBJECT_THRESHOLD</td>
<td>TINYINT</td>
<td>Displays cold_object_threshold used in the run.</td>
</tr>
<tr>
<td>MIN_OBJECT_SIZE</td>
<td>BIGINT</td>
<td>Displays min_object_size used in the run.</td>
</tr>
<tr>
<td>DURATION</td>
<td>BIGINT</td>
<td>Displays duration used in the run.</td>
</tr>
<tr>
<td>PROGRESS_DETAIL</td>
<td>NVARCHAR(5000)</td>
<td>Displays Additional information for the run.</td>
</tr>
</tbody>
</table>

### 6.2.68 M_CS_PARTITIONS System View - Deprecated

This view is deprecated. Use either the TABLE_PARTITIONS or M_CS_TABLES system view instead.

### Related Information

- TABLE_PARTITIONS System View [page 1681]
- M_CS_TABLES System View [page 1837]
6.2.69 M_CS_RECORD_LOCK_STATISTICS System View

Provides column store record lock information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ble and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is in progress.</td>
</tr>
<tr>
<td>ALLOCATED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated memory for record locks in bytes.</td>
</tr>
<tr>
<td>USED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the used memory for record locks in bytes.</td>
</tr>
<tr>
<td>ACQUIRED_LOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of locks that are currently acquired.</td>
</tr>
</tbody>
</table>

**Related Information**

HOST_RECORD_LOCKS View (Embedded Statistics Service)
HOST_OBJECT_LOCK_STATISTICS View (Embedded Statistics Service)
HOST_OBJECT_LOCKS View (Embedded Statistics Service)
### 6.2.70 M_CS_TABLE_HANDLES System View

Shows the threads waiting for table locks.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name as used in SQL.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name as used in SQL.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the unique internal table ID which remains unchanged during rename.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the logical partition number, 0 if the table is not partitioned. -1 if the logical partition number may not be retrieved from metadata.</td>
</tr>
<tr>
<td>INTERNAL_PART_ID</td>
<td>INTEGER</td>
<td>Displays the internal partition number as used in internal IndexName objects.</td>
</tr>
<tr>
<td>LOCK_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the unique identifier for the table at IndexMgr level, also found in dumps, trace output, etc.</td>
</tr>
<tr>
<td>HANDLE_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the unique memory address of the IndexHandle.</td>
</tr>
<tr>
<td>THREAD_ID</td>
<td>BIGINT</td>
<td>Displays the thread using the IndexHandle (when it was acquired).</td>
</tr>
<tr>
<td>CURRENT_STATE</td>
<td>VARCHAR(30)</td>
<td>Displays the current state of the IndexHandle. Values are: search_delta, finish_delta_merge, delete., etc. It is the lowercase state name with the prefix ns_ removed.</td>
</tr>
<tr>
<td>NEXT_STATE</td>
<td>VARCHAR(30)</td>
<td>Displays the next state of the IndexHandle when waiting for a lock upgrade. Values are the same as for CURRENT_STATE.</td>
</tr>
<tr>
<td>INITIAL_ACQUIRE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp value when IndexHandles are upgraded or downgraded to different states. This preserves the time when the IndexHandle was first acquired along with the table name.</td>
</tr>
<tr>
<td>ACQUIRE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp value for the acquired time.</td>
</tr>
</tbody>
</table>
### QUEUE_POSITION

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUEUE_POSITION</td>
<td>INTEGER</td>
<td>Displays the position within the queue, when waiting; otherwise the value is 0.</td>
</tr>
</tbody>
</table>

#### 6.2.71 M_CS_TABLE_LOCKS System View

Shows the locked tables threads waiting to lock tables.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name as used in SQL.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name as used in SQL.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the unique internal table ID, which remains unchanged during re-name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the logical partition number, 0 if the table is not partitioned, -1 if the logical partition number cannot be retrieved from metadata.</td>
</tr>
<tr>
<td>INTERNAL_PART_ID</td>
<td>INTEGER</td>
<td>Displays the internal partition number as used in internal IndexName objects.</td>
</tr>
<tr>
<td>ACQUIRED_HANDLE_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of handles acquired for the table.</td>
</tr>
<tr>
<td>WAITING_HANDLE_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of handles waiting for access to the table.</td>
</tr>
<tr>
<td>LOCK_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the unique identifier for the table at IndexMgr level, also found in dumps, trace output, etc.</td>
</tr>
<tr>
<td>IS_COORDINATOR</td>
<td>VARCHAR(5)</td>
<td>Displays the internal flag used for asserting in IndexMgr code.</td>
</tr>
<tr>
<td>IS_SHARED</td>
<td>VARCHAR(5)</td>
<td>Displays the internal flag used for asserting in IndexMgr code.</td>
</tr>
<tr>
<td>IS_BLOCK_MAIN</td>
<td>VARCHAR(5)</td>
<td>Displays the internal flag used for asserting in IndexMgr code.</td>
</tr>
<tr>
<td>IS_TABLE</td>
<td>VARCHAR(5)</td>
<td>Displays the internal flag used for asserting in IndexMgr code.</td>
</tr>
</tbody>
</table>
**IS_MODE_NO_WAIT**  
**VARCHAR(5)**  
Displays the IS_MODE_NO_WAIT flag. Access to all current handles is granted without waiting in an optimized per thread structure since none of the active handles are in conflict. This internal flag is automatically set and reset during locking operations to trigger internal performance optimizations, it does not relate to the NOWAIT keyword of SQL.

---

### 6.2.72 M_CS_TABLES System View

Provides runtime data for column tables.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Returns the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For replicated tables, the part ID is 1 for the original table and sub-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>currently in progress.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>Displays that the total memory size, in bytes, is the sum of memory size in the main, delta, and history parts.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_MAIN</td>
<td>BIGINT</td>
<td>Displays the current memory consumption, in bytes, in main. This value varies depending on the number of attributes actually loaded and includes data for open transactions.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_DELTA</td>
<td>BIGINT</td>
<td>Displays the current memory consumption, in bytes, in delta.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_HISTORY_MAIN</td>
<td>BIGINT</td>
<td>Displays the current memory consumption, in bytes, in history-main. This value is 0 for normal, non-history tables.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_HISTORY_DELTA</td>
<td>BIGINT</td>
<td>Displays the current memory consumption, in bytes, in history-delta. This value is 0 for normal non-history tables.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_PAGE_LOADABLE_MAIN</td>
<td>BIGINT</td>
<td>Displays the total paged memory size of the table.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_MISC</td>
<td>BIGINT</td>
<td>Displays the used memory size, in bytes, for the internal structures that are common to all columns in a table and that are not included in the MEMORY_SIZE_IN_MAIN or MEMORY_SIZE_IN_DELTA columns.</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>For columns stored in persistent memory, displays the total memory size in bytes.</td>
</tr>
<tr>
<td>ESTIMATED_MAX_MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>Displays the estimated maximum memory consumption, in bytes, in total, for the fully loaded table (data for open transactions is not included).</td>
</tr>
<tr>
<td>LAST_ESTIMATED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the last estimated memory consumption, in bytes, for the fully loaded table.</td>
</tr>
<tr>
<td>LAST_ESTIMATED_MEMORY_SIZE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time the last estimated memory consumption was calculated.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the record count.</td>
</tr>
<tr>
<td>RAW_RECORD_COUNT_IN_MAIN</td>
<td>BIGINT</td>
<td>Displays the current number of entries in the main part of the table. This value differs from the number of visible table main rows because there are entries for modified rows that are marked as invalidated.</td>
</tr>
<tr>
<td>RAW_RECORD_COUNT_IN_DELTA</td>
<td>BIGINT</td>
<td>Displays the current number of entries in the table delta part. This value differs from the number of visible table delta rows because there are additional entries, such as deleted rows or updated rows. This column can contain deleted records.</td>
</tr>
<tr>
<td>RAW_RECORD_COUNT_IN_HISTOR Y_MAIN</td>
<td>BIGINT</td>
<td>Displays the raw record count in history-main.</td>
</tr>
<tr>
<td>RAW_RECORD_COUNT_IN_HISTOR Y_DELTA</td>
<td>BIGINT</td>
<td>Displays the raw record count in history-delta.</td>
</tr>
<tr>
<td>LAST_COMPRESSED_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of entries in main during the last optimize compression run.</td>
</tr>
<tr>
<td>MAX_UDIV</td>
<td>BIGINT</td>
<td>Displays the maximum table row number. This number is for internal use only.</td>
</tr>
<tr>
<td>MAX_MERGE_CID</td>
<td>BIGINT</td>
<td>Displays the maximum commit-ID of transactions for which changes were already merged into the main table.</td>
</tr>
<tr>
<td>MAX_ROWID</td>
<td>BIGINT</td>
<td>Displays the maximum row ID. This number is purely technical and only used internally.</td>
</tr>
<tr>
<td>IS_DELTA2_ACTIVE</td>
<td>VARCHAR(5)</td>
<td>Indicates whether a second delta is used: TRUE/FALSE. During a table delta merge, updates and inserts are stored to a second delta because the first delta is locked.</td>
</tr>
<tr>
<td>IS_DELTA_LOADED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the delta part of the table is loaded: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_LOG_DELTA</td>
<td>VARCHAR(5)</td>
<td>Indicates whether that currently the redo log is currently being written:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE/FALSE.</td>
</tr>
<tr>
<td>PERSISTENT_MERGE</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the new main part will be written to the disk during a ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ble delta merge, unless requested differently: TRUE/FALSE.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time.</td>
</tr>
<tr>
<td>MODIFY_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the latest modify timestamp of any column store table run time data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This value is used to trigger invalidation of related translation tables used by the column store join operations.</td>
</tr>
<tr>
<td>LAST_MERGE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the point in time, in UNIX time format, of the last time the table delta part was merged into the main part.</td>
</tr>
<tr>
<td>LAST_REPLAY_LOG_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the point in time, in UNIX time format, of the last time the table log was replayed.</td>
</tr>
<tr>
<td>LAST_TRUNCATION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time the table was truncated.</td>
</tr>
<tr>
<td>LAST_CONSISTENCY_CHECK_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time the table consistency was checked with the CHECK_TABLE_CONSISTENCY procedure.</td>
</tr>
<tr>
<td>LAST_CONSISTENCY_CHECK_ERROR_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of errors found in the last table consistency check.</td>
</tr>
<tr>
<td>LOADED</td>
<td>VARCHAR(10)</td>
<td>Displays the flag to show how many columns of the table are loaded in memory: NO, PARTIALLY, and FULL. See M_CS_COLUMNS for each column.</td>
</tr>
<tr>
<td>LOAD_UNIT</td>
<td>VARCHAR(7)</td>
<td>Displays the load unit for the table: PAGE, COLUMN, and UNKNOWN.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of read accesses on the table or partition. This is not the number of SELECT statements against this table. A SELECT statement may involve several read accesses.</td>
</tr>
<tr>
<td>WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of write accesses on the table or partition. This is not the number of DML and DDL statements against this table. A DML or DDL statement may involve several write accesses.</td>
</tr>
<tr>
<td>MERGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of delta merges done on the table or partition.</td>
</tr>
<tr>
<td>IS_REPLICA</td>
<td>VARCHAR(5)</td>
<td>Displays the flag to indicate that the part is a replica.</td>
</tr>
<tr>
<td>UNUSED_RETENTION_PERIOD</td>
<td>INTEGER</td>
<td>Displays the unused retention period.</td>
</tr>
<tr>
<td>HAS_RECORD_COMMIT_TIMESTAMP</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is tracking commit timestamps: TRUE or FALSE.</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY</td>
<td>VARCHAR(5)</td>
<td>Displays the persistent memory preference inferred from user-specified preferences in bottom-up manner starting from this object: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Additional Information**

While the counter MERGE_COUNT counts merges of the table since the instance started, the corresponding LAST_MERGE_TIME timestamp persists permanently. Consequently, it is possible to see entries that have MERGE_COUNT = 0 and LAST_MERGE_TIME at some TIMESTAMP in the past. Additionally, when you copy a column store table (via CREATE COLUMN TABLE... LIKE ...) the LAST_MERGE_TIME of the original table gets copied, and the MERGE_COUNT of the new table is set to 0.

**Related Information**

M_CS_COLUMNS System View [page 1814]
TABLES System View [page 1669]
6.2.73 M_CS_UNLOADS System View

Provides a history of column unloads.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>UNLOAD_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of unload event.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name. Empty if the whole table is unloaded.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is in progress.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>IS_HISTORY</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the history part is unloaded (only relevant for history tables). Empty if the whole table is unloaded.</td>
</tr>
<tr>
<td>REASON</td>
<td>NVARCHAR(16)</td>
<td>Displays the possible reasons for the unload. Values are LOW MEMORY, SHRINK, EXPLICIT, MERGE, or UNUSED RESOURCE. Unloads caused by a manual shrink are reported as SHRINK, whereas unloads caused by memory management’s automatic shrink on out of memory (OOM) are reported as LOW MEMORY.</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY</td>
<td>VARCHAR(6)</td>
<td>Displays whether persistent memory is removed as part of the unload. Values are RETAIN or DELETE. Empty if the user did not specify a value for the unload.</td>
</tr>
</tbody>
</table>

**Related Information**

UNLOAD Statement (Data Manipulation) [page 1179]
Load/Unload a Column Table into/from Memory
Load or Unload Tables
### 6.2.74 M_CUSTOMIZABLE_FUNCTIONALITIES System View

Provides information about the enablement status of restricted features in databases. The full content of this view is only accessible from the system database.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>VARCHAR (64)</td>
<td>Displays the customizable functionality name.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR (256)</td>
<td>Displays the customizable functionality description.</td>
</tr>
<tr>
<td>IS_ENABLED</td>
<td>VARCHAR (5)</td>
<td>Indicates whether the customizable functionality is enabled: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

#### Related Information

- Restrict Features Available to a Tenant Database
- Create a Restricted Database User
- Connect to a Remote Source As a Restricted User
- Create and Authorize a Restricted User
- Disable Features on a Tenant Database
- M_FEATURES System View [page 1913]
- M_FEATURE_USAGE System View [page 1913]

### 6.2.75 M_DATABASE System View

Provides database information.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM_ID</td>
<td>VARCHAR(3)</td>
<td>Displays the system SID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the database name.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the default master host.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time.</td>
</tr>
<tr>
<td>VERSION</td>
<td>VARCHAR(32)</td>
<td>Displays the version: major.minor.patch.build.</td>
</tr>
<tr>
<td>USAGE</td>
<td>VARCHAR(32)</td>
<td>Database usage type. Recommended values: production, test, development, or custom. Any value is allowed, but recommended values may be used by clients to alter behavior.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_DATABASES System View [page 1846]
- M_DATABASE_HISTORY System View [page 1847]
- M_DATABASE_REPLICAS System View [page 1848]
- M_DATABASE_REPLICA_STATISTICS System View [page 1849]
- CREATE DATABASE Statement (Tenant Database Management) [page 746]
- ALTER DATABASE Statement (Tenant Database Management) [page 440]
- RENAME DATABASE Statement (Tenant Database Management) [page 1090]
- DROP DATABASE Statement (Tenant Database Management) [page 954]
- ALTER SYSTEM START DATABASE Statement (Tenant Database Management) [page 583]
- ALTER SYSTEM STOP DATABASE Statement (Tenant Database Management) [page 584]
- RECOVER DATABASE Statement (Backup and Recovery) [page 1070]
- Create a Database
- Tenant Databases
- Monitoring and Managing Tenant Databases
- Rename a Tenant Database
- The System Database
- Administration of Tenant Databases
- Database Details
- Manage a Database
- Database Users
- Database Roles
- Stop a Tenant Database
- Log On to a Database
## 6.2.76 M_DATABASES System View

Provides information about all databases in the system. The full content of this view is only accessible from the system database.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the database name.</td>
</tr>
<tr>
<td>ACTIVE_STATUS</td>
<td>VARCHAR (16)</td>
<td>Displays the database status.</td>
</tr>
<tr>
<td>ACTIVE_STATUS_DETAILS</td>
<td>VARCHAR (128)</td>
<td>Displays the database status details. Possible values include: stopped by user, stopped due to broken recovery, or stopped by takeover.</td>
</tr>
<tr>
<td>OS_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the database isolation operation system user.</td>
</tr>
<tr>
<td>OS_GROUP</td>
<td>NVARCHAR(256)</td>
<td>Displays the database isolation operation system group.</td>
</tr>
<tr>
<td>RESTART_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the restart behavior after a system restart. Possible values are NO (do not restart the database after a system restart) or DEFAULT (restore the database to the state it had before the system restart).</td>
</tr>
<tr>
<td>FALLBACK_SNAPSHOT_CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp for when the fallback snapshot was created, or NULL if there is no fallback snapshot for the database. The value for this column is always NULL for SYSTEMDB.</td>
</tr>
</tbody>
</table>

### Related Information

- M_DATABASE System View [page 1844]
- M_SNAPSHOTS System View [page 2134]
- M_DATABASE_HISTORY System View [page 1847]
- M_DATABASE_REPLICAS System View [page 1848]
6.2.77 M_DATABASE_HISTORY System View

Provides installation version history.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTALL_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the installation or first start time.</td>
</tr>
<tr>
<td>VERSION</td>
<td>VARCHAR(32)</td>
<td>Displays the version: major.minor.patch.build.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_DATABASE System View [page 1844]
- M_DATABASES System View [page 1846]
- PERSISTENCE_HISTORY System View [page 1608]
- Installing SAP HANA
6.2.78 M_DATABASE_REPLICAS System View

Provides source and target information for databases involved in replication.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCE_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the source database.</td>
</tr>
<tr>
<td>SOURCE_SYSTEMDB_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host of the source system database.</td>
</tr>
<tr>
<td>SOURCE_SYSTEMDB_PORT</td>
<td>INTEGER</td>
<td>Displays the source system database port.</td>
</tr>
<tr>
<td>TARGET_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the target database.</td>
</tr>
<tr>
<td>TARGET_SYSTEMDB_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the target system database host.</td>
</tr>
<tr>
<td>TARGET_SYSTEMDB_PORT</td>
<td>INTEGER</td>
<td>Displays the target system database port.</td>
</tr>
<tr>
<td>REPLICATION_STATUS</td>
<td>VARCHAR(12)</td>
<td>Displays the aggregated replication status of the database services.</td>
</tr>
</tbody>
</table>

**Related Information**

M_DATABASE_REPLICA_STATISTICS System View [page 1849]
SAP HANA System Replication with Tenant Databases
Copy or Move a Tenant Database Using Replication
M_DATABASE System View [page 1844]
6.2.79  M_DATABASE_REPLICA_STATISTICS System View

Provides statistics on databases involved in replication.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCE_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the source database.</td>
</tr>
<tr>
<td>SOURCE_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the source host name.</td>
</tr>
<tr>
<td>SOURCE_PORT</td>
<td>INTEGER</td>
<td>Displays the source internal port.</td>
</tr>
<tr>
<td>SOURCE_VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the source volume ID.</td>
</tr>
<tr>
<td>TARGET_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the target database.</td>
</tr>
<tr>
<td>TARGET_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the target host name.</td>
</tr>
<tr>
<td>TARGET_PORT</td>
<td>INTEGER</td>
<td>Displays the target port.</td>
</tr>
<tr>
<td>TARGET_ACTIVE_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the target active status.</td>
</tr>
<tr>
<td>TARGET_CONNECT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the connection was established from the target.</td>
</tr>
<tr>
<td>TARGET_RECONNECT_TIME</td>
<td>INTEGER</td>
<td>Displays the target reconnect count.</td>
</tr>
<tr>
<td>TARGET_FAILOVER_COUNT</td>
<td>INTEGER</td>
<td>Displays the target failover count.</td>
</tr>
<tr>
<td>TARGET_FULLY_RECOVERABLE</td>
<td>VARCHAR(5)</td>
<td>Indicates if the target is fully recoverable.</td>
</tr>
<tr>
<td>REPPLICATION_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the replication mode.</td>
</tr>
<tr>
<td>REPPLICATION_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the replication status.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>REPLICATION_STATUS_DETAILS</td>
<td>VARCHAR(1024)</td>
<td>Displays the replication status details.</td>
</tr>
<tr>
<td>LAST_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the current log position.</td>
</tr>
<tr>
<td>LAST_LOG_POSITION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the current log position timestamp.</td>
</tr>
<tr>
<td>SHIPPED_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the shipped log position.</td>
</tr>
<tr>
<td>SHIPPED_LOG_POSITION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the shipped log position timestamp.</td>
</tr>
<tr>
<td>SHIPPED_LOG_BUFFERS_COUNT</td>
<td>BIGINT</td>
<td>Displays the shipped log buffers count.</td>
</tr>
<tr>
<td>SHIPPED_LOG_BUFFERS_SIZE</td>
<td>BIGINT</td>
<td>Displays the shipped log buffers size in bytes.</td>
</tr>
<tr>
<td>SHIPPED_LOG_BUFFERS_DURATION</td>
<td>BIGINT</td>
<td>Displays the shipped log buffer duration in microseconds.</td>
</tr>
<tr>
<td>ASYNC_BUFFER_FULL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times the asynchronous replication buffer got full.</td>
</tr>
<tr>
<td>BACKLOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the current replication backlog in bytes.</td>
</tr>
<tr>
<td>MAX_BACKLOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the max replication backlog in bytes.</td>
</tr>
<tr>
<td>BACKLOG_TIME</td>
<td>BIGINT</td>
<td>Displays the current replication backlog in microseconds.</td>
</tr>
<tr>
<td>MAX_BACKLOG_TIME</td>
<td>BIGINT</td>
<td>Displays the max replication backlog in microseconds.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_DATABASE_REPLICAS System View [page 1848]
- M_DATABASE System View [page 1844]
- M_DATABASES System View [page 1846]
- M_DATABASE_HISTORY System View [page 1847]
- CREATE DATABASE Statement (Tenant Database Management) [page 746]
- ALTER DATABASE Statement (Tenant Database Management) [page 440]
- RENAME DATABASE Statement (Tenant Database Management) [page 1090]
- DROP DATABASE Statement (Tenant Database Management) [page 954]
- SAP HANA System Replication with Tenant Databases
Copy or Move a Tenant Database Using Replication

### 6.2.80 M_DATA_STATISTICS System View

Lists data statistics generated when you query column and row store object.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_STATISTICS_OID</td>
<td>BIGINT</td>
<td>Displays the data statistics object ID.</td>
</tr>
<tr>
<td>DATA_STATISTICS_TYPE</td>
<td>VARCHAR(12)</td>
<td>Displays the data statistics object type.</td>
</tr>
<tr>
<td>DATA_STATISTICS_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the data statistics object schema.</td>
</tr>
<tr>
<td>DATA_STATISTICS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the data statistics object name.</td>
</tr>
<tr>
<td>DATA_SOURCE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the data source.</td>
</tr>
<tr>
<td>DATA_SOURCE_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the data source object.</td>
</tr>
<tr>
<td>DATA_SOURCE_COLUMN_NAMES</td>
<td>NVARCHAR(5000)</td>
<td>Lists the column names of the data source.</td>
</tr>
<tr>
<td>DATA_SOURCE_STORAGE_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the source storage type of the data source.</td>
</tr>
<tr>
<td>DATA_SOURCE_PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID of the data source.</td>
</tr>
<tr>
<td>LAST_REFRESH_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time the data statistics on the object were last generated.</td>
</tr>
<tr>
<td>LAST_REFRESH_REASON</td>
<td>VARCHAR(19)</td>
<td>Displays the time the data statistics object was generated.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| DATA_STATISTICS_CONTENT  | NCLOB (max size 2Gb) | Lists the content and properties of the data statistics object as of the last refresh (JSON format). The content section is empty for SIMPLE, SKETCH, and RECORD COUNT; it contains bucket boundaries and counts for HISTOGRAM and a values and frequencies list for TOPK. For SAMPLE, it contains a list of sample output. The content is similar to exported data for data statistics objects.  The properties section contains properties relevant to the data statistics of the specified type. The properties section, when present, may include some of the following properties:  

**MEMORY**
Displays the memory size used by the data statistics object in bytes  
**MEMORY PERCENT**
Displays the memory size used by the data statistics object, as a percentage of the data source size  
**COUNT**
Displays the number of rows in the data source  
**DISTINCT COUNT**
Displays the number of unique values in the data source  
**NULL COUNT**
Displays the number of NULL values in the data source  
**BUCKETS**
Displays the number of buckets in the data statistics object  
**QERROR**
Displays the value of the qerror parameter used for building the data statistics object  
**QTHETA**
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>qtheta parameter</td>
<td>Displays the value of the qtheta parameter used for building the data statistics object.</td>
<td></td>
</tr>
<tr>
<td>ACCURACY</td>
<td>Displays the accuracy parameter used for building the data statistics object.</td>
<td></td>
</tr>
<tr>
<td>PREFIX BITS</td>
<td>Displays the prefix bits parameter used for building the data statistics object.</td>
<td></td>
</tr>
<tr>
<td>MIN VALUE</td>
<td>Displays the minimum value in the data source in bytes.</td>
<td></td>
</tr>
<tr>
<td>MAX VALUE</td>
<td>Displays the maximum value in the data source in bytes.</td>
<td></td>
</tr>
<tr>
<td>MAX_ROWID</td>
<td>Displays the maximum rowID for the data statistics object.</td>
<td></td>
</tr>
<tr>
<td>MIN MAX IS VALID</td>
<td>Displays whether MIN/MAX values are valid.</td>
<td></td>
</tr>
<tr>
<td>SAMPLE SIZE</td>
<td>Displays the sample size at build time, expressed in bytes.</td>
<td></td>
</tr>
<tr>
<td>SAMPLE SIZE PERCENT</td>
<td>Displays the sample size at build time, expressed as a percent of the data volume.</td>
<td></td>
</tr>
</tbody>
</table>

**Related Information**

CREATE STATISTICS Statement (Data Definition) [page 815]
ALTER STATISTICS Statement (Data Definition) [page 484]
REFRESH STATISTICS Statement (Data Definition) [page 1082]
DROP STATISTICS Statement (Data Definition) [page 974]
M_SYSTEM_DATA_STATISTICS System View [page 2189]
DATA_STATISTICS System View [page 1538]
The Statistics Service
Managing Statistics
System and Statistics Views
6.2.81 M_DATA_VOLUMES System View

Provides data volume statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>PARTITION_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is in progress.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(512)</td>
<td>Displays the filename of the data volume.</td>
</tr>
<tr>
<td>FILE_ID</td>
<td>BIGINT</td>
<td>Displays the file ID of data volume.</td>
</tr>
<tr>
<td>STATE</td>
<td>VARCHAR(16)</td>
<td>Displays the data volume state: ACTIVATING, ACTIVE, or DEACTIVATING.</td>
</tr>
<tr>
<td>SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the data volume in bytes.</td>
</tr>
<tr>
<td>MAX_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of the data volume in bytes.</td>
</tr>
</tbody>
</table>

Related Information

M_DATA_VOLUME_PAGE_STATISTICS System View [page 1855]
6.2.82  M_DATA_VOLUME_PAGE_STATISTICS System View

Provides page usage statistics on data volumes.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>PARTITION_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is in progress.</td>
</tr>
<tr>
<td>DATA_VOLUME_NAME</td>
<td>VARCHAR(512)</td>
<td>Displays the data volume name.</td>
</tr>
<tr>
<td>STATE</td>
<td>VARCHAR(16)</td>
<td>Displays the data volume state: ACTIVATING, ACTIVE, or DEACTIVATING.</td>
</tr>
</tbody>
</table>
### Additional Information

This view contains information about the number and distribution of free, used, and shadow pages inside data volumes:

- **INITIAL_BLOCK_COUNT** is the number of pages the database was started with.
- **TOTAL_*_COUNT** specifies the numbers of blocks allocated, freed or set to status FreeAfterSavepoint since the start of the database.
- **SUPERBLOCK_COUNT**, **USED_BLOCK_COUNT**, and **SHADOW_BLOCK_COUNT** columns contain the number of blocks or superblocks currently in use by the database.
- **FILL_RATIO** specifies the ratio of the minimum number of needed superblocks versus the actual number of used superblocks. Unused superblocks are not part of this formula.

This view has a resettable counterpart; you can see the values since the last reset in the **M_DATA_VOLUME_PAGE_STATISTICS_RESET** system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_DATA_VOLUME_PAGE_STATISTICS_RESET;
```
### Related Information

- M_DATA_VOLUME_PAGE_STATISTICS_RESET System View [page 1857]
- M_DATA_VOLUMES System View [page 1854]
- M_DATA_VOLUME_PARTITION_STATISTICS System View [page 1858]
- M_DATA_VOLUME_STATISTICS System View [page 1861]
- M_DATA_VOLUME_SUPERBLOCK_STATISTICS System View [page 1862]
- HOST_DATA_VOLUME_PAGE_STATISTICS View (Embedded Statistics Service)

### 6.2.83 M_DATA_VOLUME_PAGE_STATISTICS_RESET System View

Provides information about FreeBlockManager SizeClass statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_DATA_VOLUME_PAGE_STATISTICS. Refer to M_DATA_VOLUME_PAGE_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_DATA_VOLUME_PAGE_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

### Related Information

- M_DATA_VOLUME_PAGE_STATISTICS System View [page 1855]
## 6.2.84 M_DATA_VOLUME_PARTITION_STATISTICS System View

Provides data volume partition statistics.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>PARTITION_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the part ID is 1 for the original table and subsequent part IDs are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in progress.</td>
</tr>
<tr>
<td>FILE_NAME_PATTERN</td>
<td>VARCHAR(512)</td>
<td>Displays the file name pattern for the data volume partition file.</td>
</tr>
<tr>
<td>STATE</td>
<td>VARCHAR(16)</td>
<td>Displays the data volume state. This can be specified as ACTIVATING or ACT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIVE. When a data volume partition is added, it is in the ACTIVATING state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and is not usable until the next save-point, at which time it enters the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACTIVE state.</td>
</tr>
<tr>
<td>MAX_FILE_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum file size for the data volume files in bytes. A value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>of 0 means that there is no limitation for the file size.</td>
</tr>
</tbody>
</table>
### Column Descriptions

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of the data volume partition in bytes.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of the data volume partition in bytes.</td>
</tr>
<tr>
<td>FILES_SIZE</td>
<td>INTEGER</td>
<td>Displays the size of the data volume files on the disk in bytes.</td>
</tr>
<tr>
<td>FILL_RATIO</td>
<td>DOUBLE</td>
<td>Displays the fill ratio of the data volume partition.</td>
</tr>
</tbody>
</table>

### Related Information

- ALTER SYSTEM ALTER DATAVOLUME ADD PARTITION Statement (System Management) [page 503]
- ALTER SYSTEM ALTER DATAVOLUME DROP PARTITION Statement (System Management) [page 504]
- M_DATA_VOLUME_PAGE_STATISTICS_RESET System View [page 1857]
- M_DATA_VOLUMES System View [page 1854]
- M_DATA_VOLUME_STATISTICS System View [page 1861]
- M_DATA_VOLUME_SUPERBLOCK_STATISTICS System View [page 1862]
- Partitioning Data Volumes
- Static and Dynamic Partition Pruning
- Partition a Non-Partitioned Table

### 6.2.85 M_DATA_VOLUME_RECLAIM_STATISTICS System View

Displays statistical information on reclamation operations on a data volume.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the database.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the name of the Host.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>STATISTICS_ID</td>
<td>BIGINT</td>
<td>Displays the statistics object unique ID.</td>
</tr>
<tr>
<td>RECLAIM_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the reclaim status.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMPS</td>
<td>Displays the start time of the reclaim.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMPS</td>
<td>Displays the end time of the reclaim.</td>
</tr>
<tr>
<td>DURATION</td>
<td>BIGINT</td>
<td>Displays the duration of the reclaim.</td>
</tr>
<tr>
<td>RECLAIM_THRESHOLD</td>
<td>BIGINT</td>
<td>Specifies the desired percentage of the payload to which the data volume should be reduced.</td>
</tr>
<tr>
<td>STEP_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of reclaimed steps already performed.</td>
</tr>
<tr>
<td>RECLAIMED_PARTITION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of successfully reclaimed or currently reclaiming partitions.</td>
</tr>
<tr>
<td>FAILED_RECLAIM_PARTITION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of partitions on which reclaim operation failed.</td>
</tr>
<tr>
<td>PARTITION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of data volume partitions.</td>
</tr>
<tr>
<td>INITIAL_USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size in bytes before the reclaim.</td>
</tr>
<tr>
<td>INITIAL_TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size in bytes before the reclaim.</td>
</tr>
<tr>
<td>INITIAL_FILE_SIZE</td>
<td>BIGINT</td>
<td>Displays the file’s size in bytes before the reclaim.</td>
</tr>
<tr>
<td>INITIAL_FILL_RATIO</td>
<td>DOUBLE</td>
<td>Displays the fill ratio before reclaim.</td>
</tr>
<tr>
<td>RECLAIM_USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size in bytes during the reclaim or right after it.</td>
</tr>
<tr>
<td>RECLAIM_TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size in bytes during the reclaim or right after it.</td>
</tr>
<tr>
<td>RECLAIM_FILE_SIZE</td>
<td>BIGINT</td>
<td>Displays the file’s size during reclaim or right after it.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RECLAIM_FILL_RATIO</td>
<td>DOUBLE</td>
<td>Displays the fill ratio during the reclaim or right after it.</td>
</tr>
<tr>
<td>TOTAL_MOVED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size in bytes of moved data.</td>
</tr>
<tr>
<td>TOTAL_MOVED_PAGES</td>
<td>BIGINT</td>
<td>Displays the total number of moved pages.</td>
</tr>
<tr>
<td>TOTAL_MOVE_TIME</td>
<td>BIGINT</td>
<td>Displays the total time in microseconds for moving pages.</td>
</tr>
<tr>
<td>TOTAL_TRUNCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size in bytes of truncated data.</td>
</tr>
<tr>
<td>TOTAL_TRUNCATE_TIME</td>
<td>BIGINT</td>
<td>Displays the total time in microseconds for truncating data.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user name.</td>
</tr>
</tbody>
</table>

### 6.2.86 M_DATA_VOLUME_STATISTICS System View

Provides information on data volume statistics.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>PARTITION_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of partitions.</td>
</tr>
<tr>
<td>PATH_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of partition paths.</td>
</tr>
</tbody>
</table>
**Column name** | **Data type** | **Description**
--- | --- | ---
USED_SIZE | BIGINT | Displays the used size of the data volume in bytes.
TOTAL_SIZE | BIGINT | Displays the total size of the data volume in bytes.
FILL_RATIO | DOUBLE | Displays the fill ratio of the data volume.

**Related Information**

- M_DATA_STATISTICS System View [page 1851]
- M_DATA_VOLUMES System View [page 1854]
- M_DATA_VOLUME_PAGE_STATISTICS System View [page 1855]
- M_DATA_VOLUME_PARTITION_STATISTICS System View [page 1858]
- M_DATA_VOLUME_SUPERBLOCK_STATISTICS System View [page 1862]
- ALTER SYSTEM ALTER DATAVOLUME ADD PARTITION Statement (System Management) [page 503]
- ALTER SYSTEM ALTER DATAVOLUME DROP PARTITION Statement (System Management) [page 504]
- ALTER SYSTEM RECLAIM DATAVOLUME Statement (System Management) [page 551]
- M_VOLUME_SIZES System View [page 2282]

Data and Log Volumes
Partitioning Data Volumes

### 6.2.87 M_DATA_VOLUME_SUPERBLOCK_STATISTICS System View

Provides FreeBlockManager Superblock statistics.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PARTITION_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ble and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is in progress.</td>
</tr>
<tr>
<td>DATA_VOLUME_NAME</td>
<td>VARCHAR(512)</td>
<td>Displays the data volume name.</td>
</tr>
<tr>
<td>STATE</td>
<td>VARCHAR(16)</td>
<td>Displays the data volume state: ACTIVATING, ACTIVE, or DEACTIVATING.</td>
</tr>
<tr>
<td>SUPERBLOCK_SIZE</td>
<td>BIGINT</td>
<td>Displays the superblock size in bytes.</td>
</tr>
<tr>
<td>USED_SUPERBLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of used super-blocks.</td>
</tr>
<tr>
<td>SUPERBLOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of superblocks.</td>
</tr>
<tr>
<td>FILL_RATIO</td>
<td>DOUBLE</td>
<td>Displays the fill ratio.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view shows information about the number and distribution of superblocks inside data volumes:

- SUPERBLOCK_COUNT is the total number of superblocks.
- USED_SUPERBLOCK_COUNT is the number of superblocks currently occupied by at least one used or shadow page.
- FILL_RATIO specifies the ratio of the number of used superblocks versus the total number of superblocks. The fill ratio of the superblocks themselves is not part of this formula.

**Related Information**

- M_DATA_VOLUME_PAGE_STATISTICS System View [page 1855]
- M_DATA_STATISTICS System View [page 1851]
- M_DATA_VOLUMES System View [page 1854]
6.2.88 M_DEBUG_CONNECTIONS System View

Provides an overview of connections used per debug session.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBUG_SESSION_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the landscape-wide unique identifier for debug session.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection that is used for communication.</td>
</tr>
<tr>
<td>CONNECTION_USAGE</td>
<td>VARCHAR(16)</td>
<td>Indicates whether this connection is used for debugger communication or debugging.</td>
</tr>
<tr>
<td>OPERATION</td>
<td>NVARCHAR(512)</td>
<td>Displays the operation that is currently running within this connection.</td>
</tr>
</tbody>
</table>

Related Information

M_DEBUG_SESSIONS System View [page 1865]
M_CE_DEBUG_INFOS System View [page 1773]
M_CE_DEBUG_JSONS System View [page 1774]
Debug an External Session
The Debug Perspective
Debugging Procedures
Setup Debugger Privileges
Create a Debug Configuration
SQLScript Debugger
6.2.89  M_DEBUG_SESSIONS System View

Provides an overview of debug sessions and their properties.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBUG_SESSION_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the landscape-wide unique identifier for debug session.</td>
</tr>
<tr>
<td>COMPILE_MODE</td>
<td>VARCHAR(20)</td>
<td>Displays the compilation handling of nested procedures.</td>
</tr>
<tr>
<td>TIMEOUT</td>
<td>INTEGER</td>
<td>Displays the time in seconds after which the debug session will timeout and destroy itself.</td>
</tr>
<tr>
<td>ATTACH_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of debuggees the debugger is currently attached to.</td>
</tr>
<tr>
<td>ATTACH_FILTER_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID the debugger is using to attach to connections.</td>
</tr>
<tr>
<td>ATTACH_FILTER_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the connection user name the debugger is using to attach to connections.</td>
</tr>
<tr>
<td>ATTACH_FILTER_APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user name the debugger is using to attach to connections.</td>
</tr>
<tr>
<td>ATTACH_FILTER_DEBUG_TOKEN</td>
<td>VARCHAR(32)</td>
<td>Displays the debug token the debugger is using to attach to connections.</td>
</tr>
</tbody>
</table>

Related Information

- Debug Session Access
- Debug an External Session
- Debugging Procedures
- The Debug Perspective
- M_DEBUG_CONNECTIONS System View [page 1864]
- SQLScript Debugger
6.2.90 M_DELTA_MERGE_STATISTICS System View

Provides information on table delta merge statistics.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the type of the statistic:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MERGE The table delta merge.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HINT The application merge hint.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SPARSE Optimizes compression.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FACT The fact table compression.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECLAIM The table delta garbage collection.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in progress.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INTERNAL_PART_ID</td>
<td>INTEGER</td>
<td>Displays the internal partition ID.</td>
</tr>
<tr>
<td>HISTORY</td>
<td>VARCHAR(5)</td>
<td>Displays TRUE when the history part was merged. Otherwise FALSE.</td>
</tr>
<tr>
<td>MEMORY_MERGE</td>
<td>VARCHAR(5)</td>
<td>Deprecated. Use the Type column, which displays more detailed information.</td>
</tr>
<tr>
<td>PASSPORT</td>
<td>VARCHAR(256)</td>
<td>Displays the external identifier for the table merge called by an application.</td>
</tr>
<tr>
<td>LOG_REPLAY_QUEUE_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the log replay queue where the job was started. During log replay, only delta merge and optimize compression operations are possible.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the execution start time.</td>
</tr>
<tr>
<td>RESOURCE_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total wait time in milliseconds for memory and CPU resources when too many merges or optimized compressions have been started in parallel.</td>
</tr>
<tr>
<td>PHASE_1_HESITANT_LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total wait time in milliseconds to acquire an exclusive lock in the first exclusive merge phase. The time includes configured timeout and subsequent retries.</td>
</tr>
<tr>
<td>PHASE_1_BLOCKING_LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the wait time in milliseconds to acquire an exclusive lock in the first exclusive merge phase in the event the hesitant acquire was unsuccessful. New readers and writers are already blocked.</td>
</tr>
<tr>
<td>PHASE_1_LOCK_TIME</td>
<td>BIGINT</td>
<td>Displays the time in milliseconds spent under exclusive lock in the first exclusive merge phase.</td>
</tr>
<tr>
<td>PHASE_2_HESITANT_LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total wait time in milliseconds to acquire an exclusive lock for the second exclusive merge phase or the exclusive optimize compression phase, respectively. The time includes configured timeout and subsequent retries.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PHASE_2_BLOCKING_LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the time in milliseconds to acquire an exclusive lock for the second exclusive merge phase or the exclusive optimize compression phase, respectively, in the event the hesitant acquire was unsuccessful. New readers and writers are already blocked.</td>
</tr>
<tr>
<td>PHASE_2_LOCK_TIME</td>
<td>BIGINT</td>
<td>Displays the time in milliseconds spent under exclusive lock in the second exclusive merge phase or the exclusive optimize compression phase, respectively.</td>
</tr>
<tr>
<td>EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the execution duration in milliseconds.</td>
</tr>
<tr>
<td>MOTIVATION</td>
<td>VARCHAR(9)</td>
<td>Displays the motivation of the statistics:</td>
</tr>
<tr>
<td>AUTO</td>
<td></td>
<td>Triggered based on an automatic decision function.</td>
</tr>
<tr>
<td>SMART</td>
<td></td>
<td>Triggered by a HINT from the user based on a smart decision function.</td>
</tr>
<tr>
<td>CRITICAL</td>
<td></td>
<td>Triggered based on a critical decision function.</td>
</tr>
<tr>
<td>HARD</td>
<td></td>
<td>Triggered via SQL based on a hard decision function.</td>
</tr>
<tr>
<td>FORCED</td>
<td></td>
<td>A merge triggered via SQL, circumventing resource availability checks, based on a forced decision function.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SUCCESS</td>
<td>VARCHAR(5)</td>
<td>Displays a call success flag, which depends on the field TYPE.</td>
</tr>
<tr>
<td>HINT</td>
<td></td>
<td>Displays whether the application merge hint was accepted or rejected.</td>
</tr>
<tr>
<td>MERGE/SPARSE</td>
<td></td>
<td>Displays whether the delta merge/optimize compression was completed with or without success: TRUE/FALSE. For example, a table delta merge call which did not result in a delta merge because the delta was empty, is indicated with FALSE.</td>
</tr>
<tr>
<td>OLD_MAIN_RECORDS</td>
<td>INTEGER</td>
<td>Displays the number of rows in old main prior to the delta merge. -1 indicates other operation types.</td>
</tr>
<tr>
<td>MERGED_MAIN_RECORDS</td>
<td>INTEGER</td>
<td>Displays the number of rows merged from old main to new main. -1 indicates other operation types.</td>
</tr>
<tr>
<td>OLD_DELTA_RECORDS</td>
<td>INTEGER</td>
<td>Displays the number of rows merged from old main to new main. -1 indicates other operation types.</td>
</tr>
<tr>
<td>MERGED_DELTA_RECORDS</td>
<td>INTEGER</td>
<td>Displays the number of rows merged from old delta1 to new main for MERGE operation or the number of rows evicted from delta during RECLAIM delta operation. -1 indicates other operation types.</td>
</tr>
<tr>
<td>NEW_MAIN_RECORDS</td>
<td>INTEGER</td>
<td>Displays the total number of rows in new main after the merge. It is the sum of MERGED_MAIN_RECORDS and MERGED_DELTA_RECORDS. -1 indicates other operation types.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OLD_MAIN_IN_USE</td>
<td>NVARCHAR(5)</td>
<td>Displays TRUE when old main is still in use by old readers once the merge or optimize compression is finished. Old main will only be deleted after the last reader has finished. FALSE indicates other operation types.</td>
</tr>
<tr>
<td>OLD_DELTA_IN_USE</td>
<td>NVARCHAR(5)</td>
<td>Displays TRUE when old delta1 is still in use by old readers once the merge is finished. It will only be deleted after the last reader has finished. FALSE indicates other operation types.</td>
</tr>
<tr>
<td>LAST_ERROR</td>
<td>INTEGER</td>
<td>Displays the error code of the last error that occurred. This explains why a merge did not succeed. See ERROR_DESCRIPTION for details.</td>
</tr>
<tr>
<td>CS_ERROR</td>
<td>INTEGER</td>
<td>Displays the column store specific error code. See ERROR_DESCRIPTION for details.</td>
</tr>
<tr>
<td>ERROR_DESCRIPTION</td>
<td>NVARCHAR(2000)</td>
<td>Displays the description of the last error that occurred during the merge. A failing merge does not necessarily indicate a problem.</td>
</tr>
</tbody>
</table>

**Additional Information**

Table delta merges, optimize compression runs, and application merge hints are listed separately.

**Related Information**

- MERGE DELTA Statement (Data Manipulation) [page 1058]
- MERGE INTO Statement (Data Manipulation) [page 1060]
- HOST_DELTA_MERGE_STATISTICS View (Embedded Statistics Service)
- M_ES_DELTA_MERGE_STATISTICS System View [page 1889]
6.2.91 M_DISKS System View

Provides information about disk configuration and utilization of the host machine.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISK_ID</td>
<td>INTEGER</td>
<td>Displays the disk ID.</td>
</tr>
<tr>
<td>DEVICE_ID</td>
<td>BIGINT</td>
<td>Displays the database internal device ID.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name. This is only set if the disk is used by exactly one host.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the path.</td>
</tr>
<tr>
<td>SUBPATH</td>
<td>VARCHAR(512)</td>
<td>Displays the subpath.</td>
</tr>
<tr>
<td>FILESYSTEM_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the filesystem type.</td>
</tr>
<tr>
<td>USAGE_TYPE</td>
<td>VARCHAR(64)</td>
<td>Displays the usage type. Values are: LOG, DATA, TRACE, DATA_BACKUP, LOG_BACKUP, CATALOG_BACKUP, ROOTKEY_BACKUP, or XSA_WORKSPACE.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the usable total size of the device in bytes.</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of the device in bytes.</td>
</tr>
<tr>
<td>TOTAL_DEVICE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total device size returned by the operating system in bytes.</td>
</tr>
<tr>
<td>MOUNT_SOURCE</td>
<td>VARCHAR(512)</td>
<td>Displays the source of the mount.</td>
</tr>
<tr>
<td>MOUNT_PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the mount path.</td>
</tr>
<tr>
<td>MOUNTDETAILS</td>
<td>VARCHAR(2000)</td>
<td>Displays the mount details as stated in /proc/mount.</td>
</tr>
</tbody>
</table>
6.2.92 M_DISK_USAGE System View

Provides disk usage information on host basis group by resource types.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>USAGE_TYPE</td>
<td>VARCHAR(11)</td>
<td>Displays the resource type: LOG, DATA, TRACE, LOG_BACKUP or DATA_BACKUP.</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the used disk space in bytes.</td>
</tr>
</tbody>
</table>

Related Information

M_DISKSYSTEM View [page 1871]
TEL_DISK_USAGE View (Embedded Statistics Service)
Monitoring Disk Space
Disk Usage: Monitor Disk Volume
## 6.2.93 M_DSO_OPERATIONS System View

Provides information about data store object (DSO) operations.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>OPERATION</td>
<td>VARCHAR(32)</td>
<td>Displays the operation: ACTIVATION or ROLLBACK.</td>
</tr>
<tr>
<td>USAGE_MODE</td>
<td>VARCHAR(32)</td>
<td>Displays the usage mode. This clarifies which variant of DSO is used:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ADSO for advanced DSOs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ODSO for classic DSOs with persistent change log tables</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CDSO for CDS-based native DSOs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• IMO for older in-memory optimized DSOs (not recommended)</td>
</tr>
<tr>
<td>ACTIVATION_IDS</td>
<td>NVARCHAR(1024)</td>
<td>Displays the activation IDs.</td>
</tr>
<tr>
<td>TARGET_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the target table.</td>
</tr>
<tr>
<td>TARGET_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the target table.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time of the operation.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the end time of the operation.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the last error code in the case of a failure. Otherwise this value is 0.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>NVARCHAR(2000)</td>
<td>Displays the last error message in the case of a failure. Otherwise this value displays “no error”.</td>
</tr>
<tr>
<td>SOURCE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of source table rows activated.</td>
</tr>
<tr>
<td>INSERT_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows inserted into the target table.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DELETE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows deleted from the target table.</td>
</tr>
<tr>
<td>UPDATE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of target table rows that were updated.</td>
</tr>
<tr>
<td>CHANGE_LOG_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows inserted into the change log.</td>
</tr>
<tr>
<td>PACKAGE_PROCESSING_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of the processing times for all packages in microseconds.</td>
</tr>
<tr>
<td>PACKAGE_INBOUND_QUEUE_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of the inbound queue read times in all packages in microseconds.</td>
</tr>
<tr>
<td>PACKAGE_ACTIVE_DATA_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of the active data read times in all packages in microseconds.</td>
</tr>
<tr>
<td>ACTIVE_DATA_READ_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of active rows read.</td>
</tr>
<tr>
<td>PACKAGE_CALCULATION_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of the calculation times for the change log and the new active data in microseconds.</td>
</tr>
<tr>
<td>AVG_CALCULATION_JOB_COUNT</td>
<td>DOUBLE</td>
<td>Displays the active number of jobs per package used for the calculation of the change log and the new active data.</td>
</tr>
<tr>
<td>CHANGE_LOG_INSERT_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of the time spent inserting the change log records in microseconds.</td>
</tr>
<tr>
<td>ACTIVE_DATA_INSERT_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of the time spent inserting new active data records in microseconds.</td>
</tr>
<tr>
<td>ACTIVE_DATA_UPDATE_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of the time spent updating active data records in microseconds.</td>
</tr>
<tr>
<td>ACTIVE_DATA_DELETE_TIME</td>
<td>BIGINT</td>
<td>Displays the time spent deleting active data records in microseconds.</td>
</tr>
<tr>
<td>PACKAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of packages.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>NCLOB(2147483647)</td>
<td>Displays the call parameters.</td>
</tr>
</tbody>
</table>
### Configuration

**Column name** | **Data type** | **Description**  
--- | --- | ---  
CONFIGURATION | NCLOB(2147483647) | Displays the non-default, DSO-specific ini file settings used by the operation.  
USER_NAME | NVARCHAR(256) | Displays the user name.  
APPLICATION_USER_NAME | NVARCHAR(256) | Displays the application user name.

### Related Information

**Extension Node**  
M_JOB_PROGRESS System View [page 1948]  
M_EXTRACTORS System View [page 1912]

### 6.2.94 M_DYNAMIC_RESULT_CACHE System View

Lists statistics for the dynamic result cache.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACHE_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID for each dynamic result cache entry.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name that the dynamic result cache entry belongs to.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name that the dynamic result cache entry belongs to.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type that the dynamic result cache entry belongs to.</td>
</tr>
<tr>
<td>DETAILS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the details of all of the constituents of the cache key.</td>
</tr>
<tr>
<td>LAST_MVCC_SNAPSHOT_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the last cached MVCC snapshot timestamp.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the size, in bytes, of the memory occupied by the dynamic result cache entry.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_INDEX</td>
<td>BIGINT</td>
<td>Displays the size, in bytes, of the memory occupied by the dynamic result cache entry's indexes.</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of records in the cache entry.</td>
</tr>
<tr>
<td>DELTA_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of records in the cache entry created by a delta refresh.</td>
</tr>
<tr>
<td>IS_REFRESHING</td>
<td>VARCHAR(5)</td>
<td>Displays the status of the asynchronous cache entry refresh job. The result is TRUE if the refresh job is currently running and FALSE otherwise.</td>
</tr>
<tr>
<td>CURRENT_REFRESH_DURATION</td>
<td>BIGINT</td>
<td>Displays the duration, in milliseconds, of the currently running cache entry refresh job. The value is 1 when IS_REFRESHING is FALSE.</td>
</tr>
<tr>
<td>CURRENT_REFRESH_REASON</td>
<td>VARCHAR(32)</td>
<td>Displays the reason why the refresh job was triggered. The value is empty when IS_REFRESHING is FALSE.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the dynamic result cache entry was created.</td>
</tr>
<tr>
<td>LAST_REFRESH_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the last cache entry refresh job finished.</td>
</tr>
<tr>
<td>LAST_REFRESH_DURATION</td>
<td>BIGINT</td>
<td>Displays the duration, in milliseconds, of the last cache entry refresh job.</td>
</tr>
<tr>
<td>LAST_REFRESH_REASON</td>
<td>VARCHAR(32)</td>
<td>Displays the reason why the last refresh job was triggered.</td>
</tr>
<tr>
<td>LAST_DELTA_REFRESH_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the last delta refresh on the cache entry finished.</td>
</tr>
<tr>
<td>LAST_ACCESS_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of the last cache entry access.</td>
</tr>
<tr>
<td>REFRESH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of refreshes on the cache entry.</td>
</tr>
</tbody>
</table>
### Related Information

- DYNAMIC_RESULT_CACHE System View [page 1546]
- M_DYNAMIC_RESULT_CACHE_EXCLUSIONS System View [page 1877]
- DYNAMIC_RESULT_CACHE_INDEX_COLUMNS System View [page 1547]
- ALTER SYSTEM CLEAR RESULT CACHE Statement (System Management) [page 522]
- ALTER SYSTEM REMOVE RESULT CACHE ENTRY Statement (System Management) [page 561]

### 6.2.95 M_DYNAMIC_RESULT_CACHE_EXCLUSIONS System View

Lists cache exclusions of the dynamic result cache.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name that the dynamic result cache entry belongs to.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name that the dynamic result cache entry belongs to.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DETAILS</td>
<td>NVARCHAR(5000)</td>
<td>Lists all constituents of the excluded cache key.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the size, in bytes, of the memory occupied by the excluded cache entry.</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of records in the excluded cache entry.</td>
</tr>
<tr>
<td>EXCLUDE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the dynamic result cache entry was decided.</td>
</tr>
</tbody>
</table>

### Additional Information

Cache entries that exceed configurable size (defined in the `.ini` file) are automatically removed and added to `M_DYNAMIC_RESULT_CACHE_EXCLUSION`. If the view is marked to be excluded, then redefine the view definition rather than increasing the configurable size.

### Related Information

- `M_DYNAMIC_RESULT_CACHE System View [page 1875]`
- `DYNAMIC_RESULT_CACHE System View [page 1546]`
- `DYNAMIC_RESULT_CACHE_INDEX_COLUMNS System View [page 1547]`
- `ALTER SYSTEM CLEAR RESULT CACHE Statement (System Management) [page 522]`
- `ALTER SYSTEM REMOVE RESULT CACHE ENTRY Statement (System Management) [page 561]`
- Results Caching for Virtual Tables and Linked Database
### 6.2.96 M_EFFECTIVE_PASSWORD_POLICY System View

Provides information about password policy parameters for database users.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user that this password policy is valid for.</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>VARCHAR(128)</td>
<td>Displays the password policy parameter.</td>
</tr>
<tr>
<td>VALUE</td>
<td>VARCHAR(128)</td>
<td>Displays the value of the password policy parameter.</td>
</tr>
</tbody>
</table>

#### Additional Information

This view requires an equal predicate on USER_NAME.

#### Related Information

- M_PASSWORD_POLICY System View [page 2039]
- Configure Password Policies
- Password Policy Configuration Options
- Configure a Password Policy for a User Group
- CREATE USERGROUP Statement (Access Control) [page 902]
- ALTER USERGROUP Statement (Access Control) [page 665]
- GENERATE_PASSWORD Function (Security) [page 197]
- CREATE USER Statement (Access Control) [page 893]
- ALTER USER Statement (Access Control) [page 654]
- DROP USER Statement (Access Control) [page 985]
6.2.97 M_EFFECTIVE_TABLE_PLACEMENT System View

Provides information about effective placement of tables.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table.</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the record count.</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the type.</td>
</tr>
<tr>
<td>GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the group name.</td>
</tr>
<tr>
<td>GROUP_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the group type.</td>
</tr>
<tr>
<td>SUBTYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the sub type.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>VARCHAR(75)</td>
<td>Displays the location.</td>
</tr>
<tr>
<td>POSSIBLE_LOCATION</td>
<td>VARCHAR(5000)</td>
<td>Displays the possible location.</td>
</tr>
<tr>
<td>LOCATION_MATCH</td>
<td>VARCHAR(41)</td>
<td>Displays a string representing the matching rule for the possible location.</td>
</tr>
<tr>
<td>MIN_ROWS_FOR_PARTITIONING</td>
<td>BIGINT</td>
<td>Displays the minimum row count for partitioning.</td>
</tr>
<tr>
<td>MIN_ROWS_FOR_PARTITIONING_MATCH</td>
<td>VARCHAR(41)</td>
<td>Displays a string representing the matching rule for the minimum row count for partitioning.</td>
</tr>
<tr>
<td>PARTITIONING_THRESHOLD</td>
<td>BIGINT</td>
<td>Displays the repartitioning threshold.</td>
</tr>
<tr>
<td>PARTITIONING_THRESHOLD.Match</td>
<td>VARCHAR(41)</td>
<td>Displays a string representing the matching rule for the repartitioning threshold.</td>
</tr>
<tr>
<td>INITIAL_PARTITION</td>
<td>INTEGER</td>
<td>Displays the initial partitions.</td>
</tr>
<tr>
<td>INITIAL_PARTITION.Match</td>
<td>VARCHAR(41)</td>
<td>Displays a string representing the matching rule for the initial partitions.</td>
</tr>
<tr>
<td>DYNAMIC_RANGE_THRESHOLD</td>
<td>BIGINT</td>
<td>Displays the dynamic range threshold.</td>
</tr>
</tbody>
</table>
### Column Descriptions

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DYNAMIC_RANGE_THRESHOLD_MATCH</td>
<td>VARCHAR(41)</td>
<td>Displays a string representing the matching rule for the dynamic range threshold.</td>
</tr>
</tbody>
</table>

### Related Information

#### Table Placement

- Table Placement Rules
- ALTER SYSTEM ALTER TABLE PLACEMENT Statement (System Management) [page 508]
- TABLE_PLACEMENT_LOCATIONS System View [page 1685]
- TABLE_PLACEMENT System View [page 1684]

### 6.2.98 M_ENCRYPTION_OVERVIEW System View

Reports the encryption status for all data at rest where encryption is supported.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOPE</td>
<td>VARCHAR(16)</td>
<td>Displays whether the scope includes PERSISTENCE (data volumes), LOG (redo log), or BACKUP.</td>
</tr>
<tr>
<td>IS_ENCRYPTION_ACTIVE</td>
<td>VARCHAR(5)</td>
<td>Indicates if the encryption for the scope is currently active: TRUE/FALSE.</td>
</tr>
<tr>
<td>LAST_CHANGE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the latest timestamp, in the server’s local time, when the status was changed</td>
</tr>
<tr>
<td>CONFIGURATION_CONTROL</td>
<td>VARCHAR(16)</td>
<td>Indicates whether encryption configuration is controlled by the local database or the system database.</td>
</tr>
<tr>
<td>ENCRYPTION_CONTROL_LAST_CHANGE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time that the tenant encryption control was changed.</td>
</tr>
</tbody>
</table>
Related Information

ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
ALTER SYSTEM APPLICATION ENCRYPTION Statement (System Management) [page 512]
ALTER SYSTEM BACKUP ENCRYPTION Statement (System Management) [page 514]
ALTER SYSTEM ENCRYPTION CONFIGURATION Statement (System Management) [page 538]
ALTER SYSTEM LOG ENCRYPTION Statement (System Management) [page 541]
SAP HANA Backup Encryption
Encryption Configuration
Server-Side Data Encryption Services
Disable Encryption
SQLScript Encryption
Import and Export of Encrypted SQLScript Objects

6.2.99 M_EPM_SESSIONS System View

Provides all EPM sessions with detailed information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>MODEL_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the model name.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>VERSION</td>
<td>INTEGER</td>
<td>Displays the version.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(18)</td>
<td>Displays the status.</td>
</tr>
</tbody>
</table>

Related Information

EPM_QUERY_SOURCES System View [page 1560]
EPM_MODELS System View [page 1560]
Session Monitoring
6.2.100  M_ERROR_CODES System View

Provides error codes with descriptions.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE</td>
<td>INTEGER</td>
<td>Displays the error code.</td>
</tr>
<tr>
<td>CODE_STRING</td>
<td>VARCHAR(256)</td>
<td>Displays the error code string.</td>
</tr>
</tbody>
</table>

Related Information

SQL Error Codes [page 1212]
Supported Error Codes

6.2.101  M_ESgetConnections System View

Provides extended storage connection information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>INTERNAL_CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the internal connection ID.</td>
</tr>
<tr>
<td>USER_ID</td>
<td>VARCHAR(255)</td>
<td>Displays the user ID for the connection.</td>
</tr>
<tr>
<td>REQUEST_TYPE</td>
<td>VARCHAR(255)</td>
<td>Displays the last request type.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COMMAND_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of command being executed.</td>
</tr>
<tr>
<td>LOCK_OWNER_CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the connection on which this connection is blocked. This value is 0 if the connection is not blocked.</td>
</tr>
<tr>
<td>LOCK_OWNER_USER_ID</td>
<td>VARCHAR(255)</td>
<td>Displays the owner of the blocking connection. This value is NULL if there is no blocking connection.</td>
</tr>
<tr>
<td>CURSOR_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of active cursors in the connection. This value is 0 if there are no cursors.</td>
</tr>
<tr>
<td>THREAD_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of threads in the connection. The minimum value is 1.</td>
</tr>
<tr>
<td>IDLE_DURATION</td>
<td>INTEGER</td>
<td>Displays the time, in seconds, since the last SA request was issued through the connection. If there is no command, then the time since 01-01-2000 is displayed.</td>
</tr>
<tr>
<td>INTERNAL_IDLE_DURATION</td>
<td>INTEGER</td>
<td>Displays the time, in seconds, since the last command was issued through the connection. If there is no command, then the time since 01-01-2000 is displayed.</td>
</tr>
<tr>
<td>TEMP_TABLE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the temporary table space in megabytes. This value is 0 if there is no temporary table.</td>
</tr>
<tr>
<td>TEMP_WORKSPACE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the temporary workspace in megabytes. This value is 0 if there is no temporary workspace.</td>
</tr>
<tr>
<td>LAST_REQUEST_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time of the last request for the connection.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time of the connection.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the transaction ID of the current transaction on the connection.</td>
</tr>
</tbody>
</table>
Related Information

M_CONNECTIONS System View [page 1785]
CONNECT Statement (Session Management) [page 721]
Ports and Connections
Session-Specific Information for Connections
Edit an SQL Connection Configuration
SQL Connection Details

6.2.102 M_ES_DBSPACES System View

Provides extended storage dbspace information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID.</td>
</tr>
<tr>
<td>DBSPACE_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the name of the dbspace.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(11)</td>
<td>Displays the type of dbspace: MAIN, TEMPORARY, SHARED_TEMP, RLV, or CACHE.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(7)</td>
<td>Displays the status of the dbspace: OFFLINE/ONLINE.</td>
</tr>
<tr>
<td>MODE</td>
<td>VARCHAR(9)</td>
<td>Displays the dbspace mode: READONLY/READWRITE.</td>
</tr>
<tr>
<td>USAGE</td>
<td>DOUBLE</td>
<td>Displays the percent of dbspace in use.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of the dbspace in bytes.</td>
</tr>
<tr>
<td>RESERVED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total reserved space, in bytes, that can be added to dbfiles in the dbspace.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
FILE_COUNT | INTEGER | Displays the total number of files in the dbspace.
READWRITE_FILE_COUNT | INTEGER | Displays the total number of read-write files in the dbspace.
IS_DISK_STRIPING_ENABLED | VARCHAR(5) | Displays whether or not disk striping is enabled for the dbspace: TRUE/FALSE.
DISK_STRIPE_SIZE | BIGINT | Displays the disk stripe size in bytes.
IS_DROP_ALLOWED | VARCHAR(5) | Displays whether or not the dbspace can be dropped: TRUE/FALSE.

**Related Information**

M_ES_DBSPACE_FILES System View [page 1886]
ALTER EXTENDED STORAGE Statement [Dynamic Tiering] [page 1295]
Create a Space
Manage a Space
Reclaim Space
Maintain Users in a Space
Space Roles

### 6.2.103 M_ES_DBSPACE_FILES System View

Provides extended storage dbspace file information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID.</td>
</tr>
<tr>
<td>DBSPACE_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the name of the dbspace.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DBSPACE_FILE_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the name of the dbspace file.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(7)</td>
<td>Displays the status of the dbspace: OFFLINE/ONLINE.</td>
</tr>
<tr>
<td>MODE</td>
<td>VARCHAR(9)</td>
<td>Displays the dbspace mode: READONLY/READWRITE.</td>
</tr>
<tr>
<td>USAGE</td>
<td>DOUBLE</td>
<td>Displays the percent of dbspace in use.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the physical path of the file.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of the dbspace in bytes.</td>
</tr>
<tr>
<td>RESERVED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total reserved space, in bytes, that can be added to dbfiles in the dbspace.</td>
</tr>
<tr>
<td>DISK_STRIPE_SIZE</td>
<td>BIGINT</td>
<td>Displays the disk stripe size in bytes.</td>
</tr>
<tr>
<td>IS_DROP_ALLOWED</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the dbspace can be dropped: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

M_ES_DBSPACES System View [page 1885]  
ALTER EXTENDED STORAGE Statement [Dynamic Tiering] [page 1295]
Create a Space  
Manage a Space  
Reclaim Space  
Maintain Users in a Space  
Space Roles
6.2.104 M_ES_DELTA_MEMORY System View

Provides extended storage DELTA memory information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>FRAGMENT_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of store fragments for the table.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>INTEGER</td>
<td>Displays the total DELTA memory used, in megabytes, by this table.</td>
</tr>
<tr>
<td>DATA_SIZE</td>
<td>INTEGER</td>
<td>Displays the total DELTA memory used, in megabytes, to store the table data.</td>
</tr>
<tr>
<td>DICTIONARY_SIZE</td>
<td>INTEGER</td>
<td>Displays the total DELTA memory used, in megabytes, to store the dictionaries for this table.</td>
</tr>
<tr>
<td>BITMAP_SIZE</td>
<td>INTEGER</td>
<td>Displays the DELTA memory used, in megabytes, to store the table-level bitmaps.</td>
</tr>
<tr>
<td>RIDSPACE_INDEX</td>
<td>BIGINT</td>
<td>Displays the size of the RID space index in megabytes.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the associated transaction ID.</td>
</tr>
</tbody>
</table>

Related Information

M_ES_DELTA_MERGE_STATISTICS System View [page 1889]
M_ES_DELTA_MEMORY System View [Dynamic Tiering] [page 2300]
Memory Management in the Column Store
6.2.105  M_ES_DELTA_MERGE_STATISTICS System View

Provides extended storage delta merge statistics information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the merge started.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the merge ended.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the status of the merge operation: STARTED, COMPLETED, or FAILED.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the SQL code of the merge once completed.</td>
</tr>
<tr>
<td>ERROR_TEXT</td>
<td>NVARCHAR(256)</td>
<td>Displays any additional error information.</td>
</tr>
<tr>
<td>MOTIVATION</td>
<td>VARCHAR(16)</td>
<td>Displays the cause of the merge trigger: DML, DDL, SHUTDOWN, or USER.</td>
</tr>
<tr>
<td>IS_BLOCKING</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the merge was blocking: TRUE/FALSE.</td>
</tr>
<tr>
<td>INSERT_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows that were inserted as a result of the merge.</td>
</tr>
<tr>
<td>UPDATE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows that were updated as a result of the merge.</td>
</tr>
<tr>
<td>DELETE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows that were deleted as a result of the merge.</td>
</tr>
<tr>
<td>UNCOMMITTED_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows that were uncommitted at the time of the merge.</td>
</tr>
</tbody>
</table>
Related Information

M_ES_DELTA_MEMORY System View [page 1888]
MERGE DELTA Statement (Data Manipulation) [page 1058]
M_DELTA_MERGE_STATISTICS System View [page 1866]
HOST_DELTA_MERGE_STATISTICS View (Embedded Statistics Service)
The Delta Merge Operation
Perform a Delta Merge
Monitoring Delta Merge History
Perform a Manual Delta Merge Operation

6.2.106 M_ES_EXPENSIVE_STATEMENTS System View

Provides extended storage expensive statement information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction ID.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>OPERATION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the type of operation: PREPARE, EXECUTE, FETCH, CLOSE, or ES OPERATION.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the statement start time.</td>
</tr>
<tr>
<td>DURATION_MICROSEC</td>
<td>BIGINT</td>
<td>Displays the statement duration in microseconds.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>RECORDS</td>
<td>BIGINT</td>
<td>Displays the number of records.</td>
</tr>
</tbody>
</table>

Related Information

Monitor and Analyze Expensive Statements
Expensive Statements Monitoring
6.2.107 M_ES_LOCKS System View

Provides extended storage lock information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the connection ID that has the lock.</td>
</tr>
<tr>
<td>USER_ID</td>
<td>VARCHAR(255)</td>
<td>Displays the ID of the user who has the lock.</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of table: BASE TABLE, GLOBAL TEMPORARY TABLE, or MATERIALIZED VIEW.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table on which the lock is held.</td>
</tr>
<tr>
<td>INDEX_ID</td>
<td>BIGINT</td>
<td>Displays the index ID or NULL if there is no index ID.</td>
</tr>
<tr>
<td>LOCK_CLASS</td>
<td>VARCHAR(8)</td>
<td>Displays the lock class: SCHEMA, ROW, TABLE, or POSITION.</td>
</tr>
<tr>
<td>LOCK_DURATION_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of lock duration: TRANSACTION, POSITION, or CONNECTION.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOCK_TYPE</td>
<td>VARCHAR(10)</td>
<td>Displays the lock type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For SCHEMA, it could be SHARED or EXCLUSIVE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For ROW, it could be: INTENT, READPK, WRITE, WRITTENOPK, or SURROGATE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For TABLE, it could be: SHARED, INTENT, or EXCLUSIVE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For POSITION, it could be PHANTOM or INSERT.</td>
</tr>
<tr>
<td>ROW_INDENTIFIER</td>
<td>BIGINT</td>
<td>Displays the identifier for the row that the lock starts on. This value is NULL if there is no lock.</td>
</tr>
<tr>
<td>ROW_RANGE</td>
<td>BIGINT</td>
<td>Displays the number of contiguous rows that are locked. Row locks in the DELTA store can either be a single row or a range of rows.</td>
</tr>
</tbody>
</table>

**Related Information**

- LOCK TABLE Statement (Transaction Management) [page 1056]
- M_ES_LOCKS System View [Dynamic Tiering] [page 2302]
- Lock Parameters Against Editing for a Tenant Database
## 6.2.108 M_ES_OVERVIEW System View

Provides extended storage overview information.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORY</td>
<td>VARCHAR(32)</td>
<td>Displays the main category type for which the parameters and values are displayed: GENERAL, BUFFERS, MEMORY, STORAGE, DELTA, BACKUP, and VERSIONING.</td>
</tr>
<tr>
<td>NAME</td>
<td>VARCHAR(255)</td>
<td>Displays the name of the parameter provided in the VALUE column.</td>
</tr>
<tr>
<td>VALUE</td>
<td>VARCHAR(255)</td>
<td>Displays the value of the parameter.</td>
</tr>
</tbody>
</table>

### Related Information

CREATE EXTENDED STORAGE Statement [Dynamic Tiering] [page 1355]
ALTER EXTENDED STORAGE Statement [Dynamic Tiering] [page 1295]
GRANT EXTENDED STORAGE ADMIN Statement [Dynamic Tiering] [page 1407]
DROP EXTENDED STORAGE Statement [Dynamic Tiering] [page 1392]
CREATE TABLE Statement (Extended Store Table) [Dynamic Tiering] [page 1374]
ALTER TABLE Statement (Extended Store Table) [Dynamic Tiering] [page 1320]
SAP HANA Native Storage Extension
6.2.109 M_ES_RESULT_CACHE System View

Provides details about each result that has been cached. You must have the EXTENDED STORAGE ADMIN privilege to access this view.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table.</td>
</tr>
<tr>
<td>COLUMNS</td>
<td>NLCOB</td>
<td>Displays the columns included in the result as a comma-separated list.</td>
</tr>
<tr>
<td>PREDICATES</td>
<td>NCLOB</td>
<td>Displays the predicates used to generate the result as a comma-separated list.</td>
</tr>
<tr>
<td>COMMIT_ID</td>
<td>BIGINT</td>
<td>Displays the version number of the table.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the query that generated the result.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the result cache entry was generated and stored.</td>
</tr>
<tr>
<td>CREATE_DURATION</td>
<td>BIGINT</td>
<td>Displays the number of milliseconds it took to generate the original result.</td>
</tr>
<tr>
<td>CREATE_CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the connection ID that generated the result.</td>
</tr>
<tr>
<td>LAST_ACCESS_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time that the result cache entry was accessed by a user.</td>
</tr>
<tr>
<td>LAST_CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the last connection ID that accessed the result.</td>
</tr>
<tr>
<td>HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of accesses on the cache entry.</td>
</tr>
<tr>
<td>SUBSET_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of cache entry accesses that only used a subset of the result.</td>
</tr>
</tbody>
</table>
### Related Information

- ALTER SYSTEM REFRESH RESULT CACHE ENTRY Statement (System Management) [page 558]
- ALTER SYSTEM REFRESH RESULT CACHE Statement (System Management) [page 557]
- ALTER SYSTEM REMOVE RESULT CACHE ENTRY Statement (System Management) [page 561]
- ALTER SYSTEM CLEAR RESULT CACHE Statement (System Management) [page 522]
- Results Caching for Virtual Tables and Linked Database

### 6.2.110 M_ES_SERVICE_REPLICATION System View

Provides extended storage service replication information.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STORE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the name of the store.</td>
</tr>
<tr>
<td>SECONDARY_CONNECT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the connection was established on the secondary.</td>
</tr>
<tr>
<td>SECONDARY_RECONNECT_COUNT</td>
<td>INTEGER</td>
<td>Displays the secondary reconnect count.</td>
</tr>
<tr>
<td>SECONDARY_FAILOVER_COUNT</td>
<td>INTEGER</td>
<td>Displays the secondary failover count.</td>
</tr>
<tr>
<td>SECONDARY_FULLY_RECOVERABLE</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the secondary is fully recoverable.</td>
</tr>
<tr>
<td>REPLICATION_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the replication mode.</td>
</tr>
<tr>
<td>REPLICATION_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the replication status.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REPLICATION_STATUS_DETAILS</td>
<td>VARCHAR(1024)</td>
<td>Displays the replication status details.</td>
</tr>
<tr>
<td>FULL_SYNC</td>
<td>VARCHAR(16)</td>
<td>Displays the Replication Full sync status.</td>
</tr>
<tr>
<td>LAST_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the last log position.</td>
</tr>
<tr>
<td>LAST_LOG_POSITION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last log position timestamp.</td>
</tr>
<tr>
<td>SHIPPED_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the shipped log position.</td>
</tr>
<tr>
<td>SHIPPED_LOG_POSITION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the shipped log position timestamp.</td>
</tr>
<tr>
<td>SHIPPED_LOG_BUFFERS_COUNT</td>
<td>BIGINT</td>
<td>Displays the shipped log buffers count.</td>
</tr>
<tr>
<td>SHIPPED_LOG_BUFFERS_SIZE</td>
<td>BIGINT</td>
<td>Displays the shipped log buffers size in bytes.</td>
</tr>
<tr>
<td>SHIPPED_LOG_BUFFERS_DURATION</td>
<td>BIGINT</td>
<td>Displays the shipped log buffer duration in microseconds.</td>
</tr>
<tr>
<td>REPLAYED_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the log end position of the last known replayed log buffer on the secondary site.</td>
</tr>
<tr>
<td>REPLAYED_LOG_POSITION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of the last known replayed log buffer on the secondary site.</td>
</tr>
<tr>
<td>SHIPPED_FULL_REPLICA_COUNT</td>
<td>BIGINT</td>
<td>Displays the full shipped replica count.</td>
</tr>
<tr>
<td>SHIPPED_FULL_REPLICA_SIZE</td>
<td>BIGINT</td>
<td>Displays the full shipped replica size in bytes.</td>
</tr>
<tr>
<td>SHIPPED_FULL_REPLICA_DURATION</td>
<td>BIGINT</td>
<td>Displays the full shipped replica duration in microseconds.</td>
</tr>
<tr>
<td>SHIPPED_LAST_FULL_REPLICA_SIZE</td>
<td>BIGINT</td>
<td>Displays the last full shipped replica size in bytes.</td>
</tr>
<tr>
<td>SHIPPED_LAST_FULL_REPLICA_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last full shipped replica start time.</td>
</tr>
<tr>
<td>SHIPPED_LAST_FULL_REPLICA_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last full shipped replica end time.</td>
</tr>
</tbody>
</table>
### Column Name: M_ES_TABLES System View

Provides extended storage table information.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name for the table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is in progress.</td>
</tr>
<tr>
<td>IS_DELTA_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the table is an RLV table or not: TRUE/FALSE.</td>
</tr>
<tr>
<td>TABLE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the compressed data in bytes.</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the record count in the extended table.</td>
</tr>
<tr>
<td>LAST_MODIFIED_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the table was last modified.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE TABLE Statement (Extended Store Table) [Dynamic Tiering] [page 1374]
ALTER TABLE Statement (Extended Store Table) [Dynamic Tiering] [page 1320]
Managing Tables
Table Variables
## 6.2.112 M_ES_TRANSACTIONS System View

Provides extended storage transaction information.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the transaction ID.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>INTERNAL_CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the 10-digit connection ID included as a part of all messages in the esserver.trc file.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>VARCHAR(255)</td>
<td>Displays the name of the application.</td>
</tr>
<tr>
<td>USER_ID</td>
<td>VARCHAR(255)</td>
<td>Displays the user ID for the connection.</td>
</tr>
<tr>
<td>COMMIT_ID</td>
<td>BIGINT</td>
<td>Displays the ID assigned by the transaction manager when the transaction commits. For active transactions, the commit ID is 0.</td>
</tr>
<tr>
<td>VERSION_ID</td>
<td>BIGINT</td>
<td>Displays the version ID. A value of 0 indicates that the transaction has no version and that the ID has not been assigned.</td>
</tr>
<tr>
<td>TRANSACTION_STATE</td>
<td>VARCHAR(16)</td>
<td>Displays the state of the transaction: NONE, ACTIVE, ROLLING_BACK, ROLLED_BACK, COMMITTING, COMMITTED, or APPLIED.</td>
</tr>
<tr>
<td>MAIN_TABLE_USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the amount of ES_USER space created by the transaction in megabytes.</td>
</tr>
<tr>
<td>MAIN_TABLE_FREED_SIZE</td>
<td>BIGINT</td>
<td>Displays the amount of ES_USER space dropped by the transaction in megabytes.</td>
</tr>
<tr>
<td>TEMP_TABLE_USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the amount of ES_TEMP space created by the transaction in megabytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TEMP_TABLE_FREED_SIZE</td>
<td>BIGINT</td>
<td>Displays the amount of ES_TEMP space dropped by the transaction in megabytes.</td>
</tr>
<tr>
<td>TEMP_WORKSPACE_SIZE</td>
<td>BIGINT</td>
<td>For ACTIVE transactions, displays a snapshot of the work space in use at this instant by this transaction, such as sorts, hashes, and temporary bitmaps in megabytes. This number changes depending on the usage as memory is allocated and freed. When the transaction is no longer active, this column is 0.</td>
</tr>
<tr>
<td>TRANSACTION_CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that the transaction began.</td>
</tr>
<tr>
<td>CURSOR_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of open dynamic tiering cursors that reference the transaction control block. If the transaction is ACTIVE, it displays the number of open cursors created within the transaction. If the transaction is COMMITTED, it displays the number of hold cursors that reference a database version owned by the transaction.</td>
</tr>
<tr>
<td>INTERNAL_SAVEPOINT_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of savepoint structures that exist within the transaction control block.</td>
</tr>
<tr>
<td>INTERNAL_SAVEPOINT_NUMBER</td>
<td>INTEGER</td>
<td>Displays the active savepoint number of the transaction.</td>
</tr>
<tr>
<td>GLOBAL_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the global transaction ID associated with the transaction. If there is no transaction, then this value is 0.</td>
</tr>
<tr>
<td>VERSIONING_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the snapshot versioning transaction type, either table-level (default), or row-level. Row-level snapshot versioning (DELTA) applies only to DELTA-enabled tables. Once a transaction is started, this value cannot change.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_BLOCKED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not connection blocking is enabled: TRUE/FALSE. If TRUE, the transaction blocks and waits for a conflicting lock to release before it attempts to retry the lock request.</td>
</tr>
<tr>
<td>BLOCKING_TIME_OUT</td>
<td>INTEGER</td>
<td>Displays the time, in milliseconds, that a transaction waits for a locking conflict to clear. A value of 0 (default) indicates that the transaction waits indefinitely.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_ES_TRANSACTIONS System View [Dynamic Tiering] [page 2306]
- M_TRANSACTIONS System View [page 2250]
- M_BLOCKED_TRANSACTIONS System View [page 1761]
- TRANSACTION_HISTORY System View [page 1693]
- SET TRANSACTION Statement (Transaction Management) [page 1173]
- Blocked Transactions
- Blocked Transaction Monitoring

### 6.2.113 M_EVENTS System View

Provides information about internal events.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port.</td>
</tr>
<tr>
<td>SOURCE_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the original host if the event was created via another host.</td>
</tr>
<tr>
<td>SOURCE_PORT</td>
<td>INTEGER</td>
<td>Displays the original port if the event was created via another host.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
TYPE | VARCHAR(256) | Displays the type of event.
ID | BIGINT | Displays the ID of the event.
INFOTEXT | VARCHAR(2000) | Displays additional information in free text.
CREATE_TIME | TIMESTAMP | Displays the time that the event was created.
UPDATE_TIME | TIMESTAMP | Displays the time that the event was updated.
HANDLE_TIME | TIMESTAMP | Displays the time that the event was handled.
STATE | VARCHAR(256) | Displays the state of the event.
ACKNOWLEDGED | VARCHAR(5) | Indicates whether the event is acknowledged: TRUE/FALSE.
FAILED_HANDLES | BIGINT | Displays the number of failed handle attempts.

#### Additional Information

Important events (for example, DiskFull) reported by the database are shown in this view. The state of an event can be either NEW or HANDLED. CREATE_TIME shows the time the event was created (and reported) and HANDLE_TIME shows the time the event was handled (that is, cleared). Handled events are removed periodically by the StatisticsServer.

#### Related Information

- GLOBAL_INTERNAL_EVENTS View (Embedded Statistics Service)
- ALL_AUDIT_LOG System View [page 1490]
- TEL_OUT_OF_MEMORY_EVENTS View (Embedded Statistics Service)
### 6.2.114 M_EXECUTED_STATEMENTS System View

Provides all statement executions that belong to a specified category.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction object ID.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the write transaction ID. This number is ever increasing.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the unique identifier for an SQL string.</td>
</tr>
<tr>
<td>DB_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema in whose context the statement is executed.</td>
</tr>
<tr>
<td>APP_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user name.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP address of the client machine.</td>
</tr>
<tr>
<td>CLIENT_PID</td>
<td>BIGINT</td>
<td>Displays the client process ID.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the statement start time.</td>
</tr>
<tr>
<td>DURATION_MICROSEC</td>
<td>BIGINT</td>
<td>Displays the statement duration in microseconds.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the name of the related object.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the error code.</td>
</tr>
<tr>
<td>ERROR_TEXT</td>
<td>NVARCHAR(5000)</td>
<td>Displays the error message.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PASSPORT_ROOT_CONTEXT_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the SAP EPP Passport GUID that identifies the source of the request.</td>
</tr>
<tr>
<td>PASSPORT_TRANSACTION_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the SAP EPP Passport GUID that identifies the business transaction.</td>
</tr>
<tr>
<td>PASSPORT_CONNECTION_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the SAP EPP Passport GUID that identifies the connection.</td>
</tr>
<tr>
<td>PASSPORT_CONNECTION_COUNTER</td>
<td>BIGINT</td>
<td>Displays the SAP EPP Passport connection counter.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays the application that defines from which source file SAP HANA is called. The usage is up to the application. This value is also displayed in M_PREPARED_STATEMENTS.APPLICATION_SOURCE.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view allows you to collect all of the executed statements from whichever category you want to collect from. Only the DDL category is supported. The historization of DDL statement executions investigates the root cause more easily when your SAP HANA instance falls into the invalid metadata status. The collection functionality is turned off by setting the enable_ddl parameter to OFF in the [executed_statements] section of global.ini.

The collected history persists by generating the dedicated trace files (for example, indexserver_serighta3.32503.executed_statements.000.trc). You can adjust the consumed disk size for persisted trace files by configuring the maxfiles and maxfilesize parameter, which represents the maximum number of files to be generated (the default is 10) and represents the size limit of each trace file in bytes (the default is 10MB) respectively, in the [executed_statements] section of global.ini.

If the history for an executed statement is immediately persisted to the file whenever it is collected, then it may degrade the performance, as the disk input/output is usually more expensive (that is, it takes more times to flush) than memory input/output. Therefore, each statistic is stored at the in-memory buffer instead of instant disk input/output, and the buffer is flushed whenever N records are accumulated in the buffer. Adjust the degree of the flush interval with the trace_flush_interval parameter (the default is 10 records) in the [executed_statements] section of global.ini and turn OFF this feature by setting use_in_memory_tracing to OFF in the same section.

The following SQL statements are collected as DDL (Data Definition Language):

- All SQL statements starting with CREATE, DROP, ALTER, and RENAME. For example, CREATE TABLE, CREATE USER and ALTER TABLE.
• All SQL statements starting with TRUNCATE, GRANT, REVOKE, LOAD, EXPORT, IMPORT, and COMMENT.
• The SET SYSTEM LICENSE `<license_key>` and UNSET SYSTEM LICENSE ALL statements.

**Additional Information**

Setting the value of the APPLICATION_SOURCE is only available via internal APIs of the SAP HANA database client interfaces (for more information see SAP Note 2873396).

**Related Information**

SAP Note 2873396
Controlling Parallel Execution of SQL Statements
Execute SQL Statements in SAP HANA Studio
SQL Statements [page 430]
EXECUTE IMMEDIATE
Explicit Parallel Execution

**6.2.115 M_EXPENSIVE_STATEMENTS System View**

Provides all statements with a duration longer than a specified threshold.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction object ID.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the write transaction ID.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>NVARCHAR(256)</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>PARENT_STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the parent statement ID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the unique identifier for an SQL string.</td>
</tr>
<tr>
<td>DB_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema in whose context the statement is executed.</td>
</tr>
<tr>
<td>APP_USER</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user name.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the statement start time.</td>
</tr>
<tr>
<td>DURATION_MICROSEC</td>
<td>BIGINT</td>
<td>Displays the statement duration in microseconds.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the related objects.</td>
</tr>
<tr>
<td>OPERATION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the type of operation (prepare, execute, fetch, or close).</td>
</tr>
<tr>
<td>RECORDS</td>
<td>BIGINT</td>
<td>Displays the number of records.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the statement parameters.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the error code.</td>
</tr>
<tr>
<td>ERROR_TEXT</td>
<td>NVARCHAR(5000)</td>
<td>Displays the error message.</td>
</tr>
<tr>
<td>LOCK_WAIT_COUNT</td>
<td>INTEGER</td>
<td>Displays the accumulated lock wait count.</td>
</tr>
<tr>
<td>LOCK_WAIT_DURATION</td>
<td>BIGINT</td>
<td>Displays the accumulated lock wait duration in microseconds.</td>
</tr>
<tr>
<td>ALLOC_MEM_SIZE_ROWSTORE</td>
<td>BIGINT</td>
<td>Deprecated, do not use.</td>
</tr>
<tr>
<td>ALLOC_MEM_SIZE_COLSTORE</td>
<td>BIGINT</td>
<td>Deprecated, do not use.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the peak memory usage in bytes during the execution of the statement. In cases where a query was executed in a scale-out system (i.e. the query was running distributed across multiple hosts), then MEMORY_SIZE is the highest peak memory usage across all involved hosts. So if host A had a peak usage of 2 GB and host B has a peak usage of 3 GB, then MEMORY_SIZE displays 3 GB. This value is kept for cache reasons or released after the statement execution.</td>
</tr>
<tr>
<td>REUSED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory that is reused from cached data structures in bytes. The value is 0 if the cache is empty.</td>
</tr>
<tr>
<td>CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the CPU time, in microseconds, consumed to compute the statement.</td>
</tr>
<tr>
<td>PASSPORT_ROOT_CONTEXT_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the SAP EPP Passport GUID that identifies the source of the request.</td>
</tr>
<tr>
<td>PASSPORT_TRANSACTION_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the SAP EPP Passport GUID that identifies the business transaction.</td>
</tr>
<tr>
<td>PASSPORT_CONNECTION_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the SAP EPP Passport GUID that identifies the connection.</td>
</tr>
<tr>
<td>PASSPORT_CONNECTION_COUNTER</td>
<td>BIGINT</td>
<td>Displays the SAP EPP Passport connection counter.</td>
</tr>
<tr>
<td>STATEMENT_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time when the first operation of a statement execution starts. All operations of a statement execution (prepare, execute, fetch, or close) share the same statement start time.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays the application from which source file SAP HANA is called. Usage is determined by the application. The value is also displayed in M_PREPARED_STATEMENTS.APPLICATION_SOURCE.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NETWORK_MESSAGE_ID</td>
<td>BIGINT</td>
<td>Displays the client message ID on the physical server connection, also found in M_SQL_CLIENT_NETWORK_IO and MESSAGE_ID.</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the effective workload class.</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>INTEGER</td>
<td>Displays the effective statement priority.</td>
</tr>
<tr>
<td>STATEMENT_THREAD_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective statement thread limit.</td>
</tr>
<tr>
<td>STATEMENT_MEMORY_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective statement memory limit.</td>
</tr>
<tr>
<td>SESSION_VARIABLES</td>
<td>NCLOB</td>
<td>Displays the statement’s session variables.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_MEMORY_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective total statement memory limit.</td>
</tr>
</tbody>
</table>

**Additional Information**

Allows you to monitor all statements that have an execution time longer than a predefined threshold (configurable in the [expensive_statement] section of the global.ini file), which may sometimes cause barriers.

The M_EXPENSIVE_STATEMENTS view provides convenient access to the most expensive statements that are executed on the system. A statement is considered expensive if its runtime exceeds a particular threshold.

The following types of operations are tracked:

- **COMPILE**: compiles the statement.
- **FETCH**: fetches the result set.
- **CURSOR_CLOSE**: the closing cursor.
- **AGGREGATED_EXECUTION**: calculates the overall execution time for SELECT queries, including compilation time in the case of non-prepared statements.
  For prepared statements, compilation time is not included in the AGGREGATED_EXECUTION of every prepared statement execution, since compilation usually occurs only once (except for recompilation at runtime). Adding compilation time to AGGREGATED_EXECUTION may exaggerate the overall execution time.
  For non-prepared statements, compilation time is included in the AGGREGATED_EXECUTION of every ad hoc statement execution since it is newly compiled or is attempting to look up a SQL plan cache every time within a single session request.
The following types of operations execute a DML statement:

- CALL
- DELETE
- DEQUEUE
- EXPLAIN_PLAN
- INSERT
- SELECT
- SELECT_FOR_UPDATE
- UPDATE
- EXECUTE_DML: represents the less common DML statements that are not listed above.

The following operations execute a DDL statement:

- CREATE_TABLE
- CREATE_VIEW
- EXECUTE_DDL: represents the less common DDL statements that are not listed above.

The following operations execute statements except DML and DDL:

- EXECUTE: represents the less common SQL statements.

Consider a query with 4 session requests from a client:

1. Prepare: 10 msec
2. Execute: 20 msec
3. Fetch: 15 msec
4. Close: 5 msec

Assuming that all operations are longer than the threshold, they are shown as: OPERATION DURATION

COMPILE 10 SELECT 20 FETCH 15 CURSOR_CLOSE 5 AGGREGATED_EXECUTION 40 (20+15+5).

Additional Information

Setting the value of the APPLICATION_SOURCE is only available via internal APIs of the SAP HANA database client interfaces (for more information see SAP Note 2873396).

Related Information

SAP Note 2873396
M_EXPENSIVE_STATEMENT_EXECUTION_LOCATION_STATISTICS System View [page 1910]
Monitor and Analyze Expensive Statements
Expensive Statements Monitoring
Expensive Statements Trace
6.2.116 M_EXPENSIVE_STATEMENT_EXECUTION_LOCATION_STATISTICS System View

Provides location statistics for expensive statements.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>STATEMENT_EXECUTION_ID</td>
<td>BIGINT</td>
<td>Displays the execution ID of the statement.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the execution start time of the statement.</td>
</tr>
<tr>
<td>EXECUTION_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where the statement was executed.</td>
</tr>
<tr>
<td>EXECUTION_PORT</td>
<td>INTEGER</td>
<td>Displays the port of the host where the statement was executed.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the peak memory usage for the host in bytes. For example, if host A has a peak usage of 2 GB and host B has a peak usage of 3 GB, then there is a row with host A showing 2 GB and a row with host B showing 3 GB.</td>
</tr>
<tr>
<td>CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the tracked CPU time that each location consumed.</td>
</tr>
</tbody>
</table>

**Related Information**

M_EXPENSIVE_STATEMENTS System View [page 1905]
Monitor and Analyze Expensive Statements
Expensive Statements Monitoring
Expensive Statements Trace
6.2.117 M_EXPORT_BINARY_STATUS System View

Provides export status information for the current session.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID to distinguish the data of each export session</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema of the table or view being exported.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table or view being exported.</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the index type: UNKNOWN, PHYSICAL, OLAP, JOIN, HIERARCHY, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CALCULATION.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the export status: QUEUED, WORKING, SKIPPED, DONE, or FAILED.</td>
</tr>
<tr>
<td>ERROR</td>
<td>VARCHAR(512)</td>
<td>Displays empty if the export is successful. If the export was unsuccessful,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>then this row displays error text.</td>
</tr>
</tbody>
</table>

Related Information

- EXPORT Statement (Data Import Export) [page 997]
- IMPORT Statement (Data Import Export) [page 1029]
- Binary Data Types [page 35]
- Export Schemas, Tables, and Other Catalog Objects
- Export Tables and Other Catalog Objects
6.2.118  M_EXTRACTORS System View

Provides direct extractor connection (DXC) status information

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema where the DSO tables are generated.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the DSO-specific status table.</td>
</tr>
<tr>
<td>CLIENT</td>
<td>NVARCHAR(3)</td>
<td>Displays the ID of the client.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(5)</td>
<td>Displays the general status overview: OK/ERROR.</td>
</tr>
<tr>
<td>REQUEST_ID</td>
<td>INTEGER</td>
<td>Displays the SID of the request used for activation.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the message in case of an error.</td>
</tr>
<tr>
<td>PHASE</td>
<td>VARCHAR(32)</td>
<td>Displays the phase of the last running operation.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of the operation phase.</td>
</tr>
</tbody>
</table>

Related Information

EXTRACT Function (Datetime) [page 194]
Data Access
GLOBAL_DEC_EXTRACTOR_STATUS View (Embedded Statistics Service)
6.2.119 M_FEATURES System View

Provides information about all supported features.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPONENT_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the name of the component owning the feature.</td>
</tr>
<tr>
<td>FEATURE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the name of the feature.</td>
</tr>
<tr>
<td>FEATURE_VERSION</td>
<td>BIGINT</td>
<td>Displays the version of the feature.</td>
</tr>
</tbody>
</table>

**Related Information**

M_FEATURE_USAGE System View [page 1913]
Restrict Features Available to a Tenant Database
Disable Features on a Tenant Database

6.2.120 M_FEATURE_USAGE System View

Provides detailed feature usage statistics.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPONENT_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the component name.</td>
</tr>
<tr>
<td>FEATURE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the feature name.</td>
</tr>
<tr>
<td>IS_DEPRECATED</td>
<td>VARCHAR(5)</td>
<td>When set to TRUE, displays if a feature is deprecated, otherwise it is FALSE.</td>
</tr>
</tbody>
</table>

SAP HANA SQL Reference Guide for SAP HANA Platform
System Views Reference
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_COUNT</td>
<td>BIGINT(19)</td>
<td>Displays the counter tracking the number of artifacts that belong to the feature. For example, the number of existing history tables.</td>
</tr>
<tr>
<td>CALL_COUNT</td>
<td>BIGINT(19)</td>
<td>Displays the counter tracking the number of times the feature was used since the last start of the index server. For example, number of accesses to any history table.</td>
</tr>
<tr>
<td>LAST_TIMESTAMP</td>
<td>TIMESTAMP(27)</td>
<td>Displays the last point in time when the feature was used. By default, the value is deprecated if this configuration has not changed.</td>
</tr>
<tr>
<td>LAST_CONNECTION_ID</td>
<td>INTEGER(10)</td>
<td>Displays the connection ID of the last feature usage. By default, the value is deprecated if this configuration has not changed.</td>
</tr>
<tr>
<td>LAST_USER_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the database user of the last feature usage, for example, SAP &lt;SID&gt;. By default, the value is deprecated if this configuration has not changed.</td>
</tr>
<tr>
<td>LAST_APPLICATION_USER_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the application user of the last feature usage, for example, &lt;D-User&gt;. By default, the value is deprecated if this configuration has not changed.</td>
</tr>
<tr>
<td>LAST_APPLICATION_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the application name of the last feature usage. By default, the value is deprecated if this configuration has not changed.</td>
</tr>
<tr>
<td>LAST_APPLICATION_SOURCE</td>
<td>VARCHAR(256)</td>
<td>Displays the application source name and location of the last feature usage. By default, the value is deprecated if this configuration has not changed.</td>
</tr>
<tr>
<td>LAST_STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the statement ID of the last feature usage. By default, the value is deprecated if this configuration has not changed.</td>
</tr>
</tbody>
</table>
### Column Name Description

| LAST_STATEMENT_HASH | VARCHAR(32) | Displays the statement hash of the last feature usage. By default, the value is deprecated if this configuration has not changed. |

### Native Storage Extension Feature Monitoring

Native Storage Extension (NSE) feature usage statistics can be seen in M_FEATURE_USAGE monitoring view.

Under the component name NSE, object count for table, column, partition and index are shown.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_COUNT here indicates the number of objects (table, column, partition, index) which have load unit defined as page-loadable.</td>
</tr>
</tbody>
</table>

### Examples

**TABLE**

<table>
<thead>
<tr>
<th>COMPONENT_NAME</th>
<th>FEATURE_NAME</th>
<th>IS_DEPRECATED</th>
<th>OBJECT_COUNT</th>
<th>OTHER COLUMNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSE</td>
<td>TABLE</td>
<td>FALSE</td>
<td>10</td>
<td>?</td>
</tr>
</tbody>
</table>

**COLUMN**

<table>
<thead>
<tr>
<th>COMPONENT_NAME</th>
<th>FEATURE_NAME</th>
<th>IS_DEPRECATED</th>
<th>OBJECT_COUNT</th>
<th>OTHER COLUMNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSE</td>
<td>COLUMN</td>
<td>FALSE</td>
<td>12</td>
<td>?</td>
</tr>
</tbody>
</table>

**PARTITION**

<table>
<thead>
<tr>
<th>COMPONENT_NAME</th>
<th>FEATURE_NAME</th>
<th>IS_DEPRECATED</th>
<th>OBJECT_COUNT</th>
<th>OTHER COLUMNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSE</td>
<td>PARTITION</td>
<td>FALSE</td>
<td>15</td>
<td>?</td>
</tr>
</tbody>
</table>

**INDEX**

<table>
<thead>
<tr>
<th>COMPONENT_NAME</th>
<th>FEATURE_NAME</th>
<th>IS_DEPRECATED</th>
<th>OBJECT_COUNT</th>
<th>OTHER COLUMNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSE</td>
<td>INDEX</td>
<td>FALSE</td>
<td>2</td>
<td>?</td>
</tr>
</tbody>
</table>

### Related Information

- **M_FEATURES System View [page 1913]**
- *Restrict Features Available to a Tenant Database*
6.2.121 M_FULLTEXT_COLUMN_STATISTICS System View

Tracks the usage of fulltext indexes.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name to which the table belongs.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the internal name of the text column.</td>
</tr>
<tr>
<td>LOOKUP_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of text look-ups executed on this column.</td>
</tr>
<tr>
<td>LAST_LOOKUP_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the last lookup was executed.</td>
</tr>
</tbody>
</table>

**Related Information**

M_FULLTEXT_QUEUES System View [page 1917]
CREATE FULLTEXT INDEX Statement (Data Definition) [page 748]
ALTER FULLTEXT INDEX Statement (Data Definition) [page 445]
DROP FULLTEXT INDEX Statement (Data Definition) [page 955]
### 6.2.122 M_FULLTEXT_QUEUES System View

Provides fulltext index queue status.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>INDEX_OID</td>
<td>BIGINT</td>
<td>Displays the OID of the corresponding fulltext index.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema of the data table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the data table.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ble and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is in progress.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the document column.</td>
</tr>
<tr>
<td>STATUS</td>
<td>NVARCHAR(16)</td>
<td>Displays the queue status: ACTIVE/SUSPENDED.</td>
</tr>
<tr>
<td>TOTAL_DOCUMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of documents in the original table.</td>
</tr>
<tr>
<td>INDEXED_DOCUMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of successfully indexed documents.</td>
</tr>
<tr>
<td>QUEUE_DOCUMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of documents currently in the queue.</td>
</tr>
</tbody>
</table>
**Column name**  
**Data type**  
**Description**  

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_DOCUMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of failed documents.</td>
</tr>
</tbody>
</table>

**Related Information**

*M_FULLTEXT_COLUMN_STATISTICS System View* [page 1916]  
*ALTER FULLTEXT INDEX Statement (Data Definition)* [page 445]

**6.2.123  M_FUZZY_SEARCH_INDEXES System View**

Provides runtime information of fuzzy search indexes of column tables.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is in progress.</td>
</tr>
<tr>
<td>FUZZY_SEARCH_MODE</td>
<td>NVARCHAR(256)</td>
<td>Displays the fuzzy index search mode.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>Displays the sum of MEMORY_SIZE_IN_MAIN and MEMORY_SIZE_IN_DELTA in bytes.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_MAIN</td>
<td>BIGINT</td>
<td>Displays the current size of the fuzzy search indexes in main in bytes. This value is 0 if not loaded.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_DELTA</td>
<td>BIGINT</td>
<td>Displays the current size of the fuzzy search indexes in delta, in bytes. This value is 0 if not loaded.</td>
</tr>
<tr>
<td>LOADED</td>
<td>VARCHAR(5)</td>
<td>Displays a flag to indicate that the column is loaded into memory: TRUE/ FALSE.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view shows the memory usage of all fuzzy search indexes. A fuzzy search index has no name and is identified by its schema name, table name, and column name.

**Related Information**

CREATE FULLTEXT INDEX Statement (Data Definition) [page 748]
ALTER FULLTEXT INDEX Statement (Data Definition) [page 445]
INDEXES System View [page 1585]
## 6.2.124  M_GARBAGE_COLLECTION_STATISTICS System View

Provides garbage collection and history manager statistics.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>STORE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of storage handled: COLUMN STORE/LIVECACHE.</td>
</tr>
<tr>
<td>HISTORY_COUNT</td>
<td>INTEGER</td>
<td>Displays the current count of history files in the garbage collection.</td>
</tr>
<tr>
<td>WAITER_COUNT</td>
<td>INTEGER</td>
<td>Displays the current count of garbage collection waiters.</td>
</tr>
<tr>
<td>MIN_READ_TID</td>
<td>BIGINT</td>
<td>Displays the last known minimum transaction ID (TID) of a reading transaction at end of transaction (EOT).</td>
</tr>
<tr>
<td>LAST_STARTED_TID</td>
<td>BIGINT</td>
<td>Displays the TID of the last started cleanup job.</td>
</tr>
<tr>
<td>LAST_STARTED_TID_POSTCOMMIT</td>
<td>BIGINT</td>
<td>Displays the TID of the last started postcommit job.</td>
</tr>
<tr>
<td>FIRST_WAITING_TID</td>
<td>BIGINT</td>
<td>Displays the TID of the first waiting cleanup job.</td>
</tr>
<tr>
<td>FIRST_WAITING_TID_POSTCOMMIT</td>
<td>BIGINT</td>
<td>Displays the TID of the first waiting postcommit job.</td>
</tr>
<tr>
<td>ENTERS</td>
<td>BIGINT</td>
<td>Displays the number of history files that entered the queue for cleanup.</td>
</tr>
<tr>
<td>ENTERS_POSTCOMMIT</td>
<td>BIGINT</td>
<td>Displays the number of history files that entered the queue for postcommit.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STARTED_JOBS</td>
<td>BIGINT</td>
<td>Displays the number of started garbage collection cleanup jobs.</td>
</tr>
<tr>
<td>STARTED_JOBS_POSTCOMMIT</td>
<td>BIGINT</td>
<td>Displays the number of started post-commit jobs.</td>
</tr>
<tr>
<td>PROCESSED_JOBS</td>
<td>BIGINT</td>
<td>Displays the number of undo files processed for cleanup.</td>
</tr>
<tr>
<td>PROCESSED_JOBS_POSTCOMMIT</td>
<td>BIGINT</td>
<td>Displays the number of undo files processed for postcommit.</td>
</tr>
<tr>
<td>QUEUE_LOADS</td>
<td>BIGINT</td>
<td>Displays the number of all garbage collection queue loads.</td>
</tr>
<tr>
<td>QUEUE_LOADS_NONEMPTY</td>
<td>BIGINT</td>
<td>Displays the number of garbage collection queue loads which found elements.</td>
</tr>
<tr>
<td>QUEUE_EMPTY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of queue empty states after garbage collection finished.</td>
</tr>
<tr>
<td>SAVEPOINTS</td>
<td>BIGINT</td>
<td>Displays the number of savepoints.</td>
</tr>
<tr>
<td>LAST_HISTORY_SIZE_AT_SVP</td>
<td>BIGINT</td>
<td>Displays the lastest number of history files present at the savepoint.</td>
</tr>
<tr>
<td>MAX_HISTORY_SIZE_AT_SVP</td>
<td>BIGINT</td>
<td>Displays the maximum number of history files present at the savepoint.</td>
</tr>
<tr>
<td>MIN_HISTORY_SIZE_AT_SVP</td>
<td>BIGINT</td>
<td>Displays the minimum number of history files present at the savepoint.</td>
</tr>
<tr>
<td>SUM_HISTORY_SIZE_AT_SVP</td>
<td>BIGINT</td>
<td>Displays the total number of history files present at the savepoint.</td>
</tr>
<tr>
<td>AVG_HISTORY_SIZE_AT_SVP</td>
<td>DOUBLE</td>
<td>Displays the average number of history files present at the savepoint.</td>
</tr>
<tr>
<td>EMPTY_HISTORY_AT_SVP_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of savepoints with an empty queue.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view shows various statistics about garbage collection jobs.

Garbage collection is used to remove old versions of data objects from the system. Afterward, transactions cannot reference these old versions. References to these objects are kept in history (cleanup) files, which are processed by the garbage collector.
Normally, the system runs in statement consistency isolation level (that is, a consistent view is acquired when an SQL operation starts). Alternatively, the system can be run in transaction consistency isolation level, where the consistent view is acquired when the transaction starts and is held until the transaction terminates (snapshot isolation, similar to SERIALIZABLE isolation level). Each consistent view contains a reference to a minReadTID Transaction ID (TID), which is the minimum TID, from which the changes are "seen" by the transaction. Global minReadTID (minimum of minReadTIDs of all consistent views) is used by the garbage collector to determine which history files can be cleaned.

In cases where the history files accumulate and MIN_READ_TID does not advance, some transactions probably hold their consistent view for very long time (for example, a long-running transaction of a forgotten read transaction in a GUI). Query the M_TRANSACTIONS system view to see information about each running transaction.

This view has a resettable counterpart; you can see the values since the last reset in the M_GARBAGE_COLLECTION_STATISTICS_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_GARBAGE_COLLECTION_STATISTICS_RESET;
```

### Related Information

- **M_GARBAGE_COLLECTION_STATISTICS_RESET System View** [page 1922]
- **M_TRANSACTIONS System View** [page 2250]
- **ALTER SYSTEM RECLAIM VERSION SPACE Statement (System Management)** [page 554]
- **ALTER SYSTEM RESET MONITORING VIEW Statement (System Management)** [page 555]
- **ALTER SYSTEM RECLAIM FLEXIBLE TABLES Statement (System Management)** [page 552]

### 6.2.125 M_GARBAGE_COLLECTION_STATISTICS_RESET System View

Provides accumulated garbage collection and history manager statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_GARBAGE_COLLECTION_STATISTICS. Refer to M_GARBAGE_COLLECTION_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_GARBAGE_COLLECTION_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

### Related Information

- **M_GARBAGE_COLLECTION_STATISTICS System View** [page 1920]
6.2.126 M_HA_DR_PROVIDERS System View

Provides information about loaded HA/DR providers.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROVIDER_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the HA/DR provider name.</td>
</tr>
<tr>
<td>PROVIDER_COMPANY</td>
<td>VARCHAR(64)</td>
<td>Displays the HA/DR provider company name.</td>
</tr>
<tr>
<td>PROVIDER_VERSION</td>
<td>VARCHAR(32)</td>
<td>Displays the HA/DR provider version.</td>
</tr>
<tr>
<td>PROVIDER_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the HA/DR provider type.</td>
</tr>
<tr>
<td>PROVIDER_PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the HA/DR provider path.</td>
</tr>
<tr>
<td>EXECUTION_ORDER</td>
<td>INTEGER</td>
<td>Displays the HA/DR provider execution order.</td>
</tr>
</tbody>
</table>

Related Information

Implementing a HA/DR Provider
Create a HA/DR Provider
Install and Configure a HA/DR Provider Script
Example HA/DR Provider Implementation
6.2.127 M_HEAP_MEMORY System View

Provides memory allocator statistics.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>STATISTICS_ID</td>
<td>BIGINT</td>
<td>Displays the statistics object unique ID.</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>VARCHAR(128)</td>
<td>Displays the allocator name.</td>
</tr>
<tr>
<td>DEPTH</td>
<td>BIGINT</td>
<td>Displays the depth.</td>
</tr>
<tr>
<td>INCLUSIVE_SIZE_IN_USE</td>
<td>BIGINT</td>
<td>Displays the current size of this allocator, including suballocators in bytes.</td>
</tr>
<tr>
<td>INCLUSIVE_COUNT_IN_USE</td>
<td>BIGINT</td>
<td>Displays the number of blocks currently in use, including suballocators.</td>
</tr>
<tr>
<td>INCLUSIVE_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total allocated size in this allocator and suballocators in bytes.</td>
</tr>
<tr>
<td>INCLUSIVE_DEALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total deallocated size in this allocator and suballocators in bytes.</td>
</tr>
<tr>
<td>INCLUSIVE_ALLOCATED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of allocations, including suballocators.</td>
</tr>
<tr>
<td>INCLUSIVE_DEALLOCATED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deallocations, including suballocators.</td>
</tr>
<tr>
<td>INCLUSIVE_MAX_SINGLE_ALLOCATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest block size in bytes ever allocated in this allocator and suballocators.</td>
</tr>
<tr>
<td>INCLUSIVE_PEAK_ALLOCATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest size in bytes of this allocator and suballocators (estimate).</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------************************************************************************</td>
</tr>
<tr>
<td>INCLUSIVE_IN_USE_INTEGRAL</td>
<td>BIGINT</td>
<td>Displays the average allocated memory in bytes by this allocator and its suballocators, multiplied by the time since the start of measurement (sample based rough estimate). Deactivated by default this value should only be activated upon request by SAP support. UOM is 1 byte times 1 second.</td>
</tr>
<tr>
<td>EXCLUSIVE_SIZE_IN_USE</td>
<td>BIGINT</td>
<td>Displays the current size of this allocator in bytes.</td>
</tr>
<tr>
<td>EXCLUSIVE_COUNT_IN_USE</td>
<td>BIGINT</td>
<td>Displays the number of blocks currently in use.</td>
</tr>
<tr>
<td>EXCLUSIVE_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total allocated size in this allocator in bytes.</td>
</tr>
<tr>
<td>EXCLUSIVE_DEALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total deallocated size in this allocator in bytes.</td>
</tr>
<tr>
<td>EXCLUSIVE_ALLOCATED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of allocations.</td>
</tr>
<tr>
<td>EXCLUSIVE_DEALLOCATED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deallocations.</td>
</tr>
<tr>
<td>EXCLUSIVE_MAX_SINGLE_ALLOCATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest block size in bytes ever allocated in this allocator.</td>
</tr>
<tr>
<td>EXCLUSIVE_PEAK_ALLOCATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest size of this allocator (estimate) in bytes.</td>
</tr>
<tr>
<td>EXCLUSIVE_ALLOC_ERRORS</td>
<td>BIGINT</td>
<td>Displays the number of allocation errors.</td>
</tr>
<tr>
<td>EXCLUSIVE_IN_USE_INTEGRAL</td>
<td>BIGINT</td>
<td>Displays the average allocated memory in bytes by this allocator, multiplied by time since start of measurement (sample based rough estimate). Deactivated by default, this value should only be activated upon request by SAP support. UOM is 1 byte times 1 second.</td>
</tr>
<tr>
<td>MALLOC_PROXY_CACHE_MISSES</td>
<td>BIGINT</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>FLAGS</td>
<td>VARCHAR(64)</td>
<td>Displays the allocator flags.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VARCHAR(64)</td>
<td>Displays the name of SAP HANA component of this allocator.</td>
</tr>
</tbody>
</table>
**Additional Information**

This view contains information about memory consumption of various components in the system. Parallel to heap memory, you can also query memory consumption by connection, statement, and user using the `M_CONTEXT_MEMORY` system view.

The overhead of allocators is not considered in this view.

This view has a resettable counterpart; you can see the values since the last reset in the `M_HEAP_MEMORY_RESET` system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_HEAP_MEMORY_RESET;
```

**Related Information**

- `M HEAP MEMORY RESET System View [page 1927]`
- `M HEAP MEMORY AREAS System View [page 1926]`
- `M CONTEXT MEMORY System View [page 1801]`
- `M SERVICE MEMORY System View [page 2113]`

**6.2.128 M_HEAP_MEMORY AREAS System View**

Provides memory fragmentation details.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>AREA</td>
<td>STRING</td>
<td>Identifies the part of the memory management being used, for example, the PoolAllocator.</td>
</tr>
<tr>
<td>USED_SIZED</td>
<td>BIGINT</td>
<td>Displays the bytes in use.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the total bytes in the area.</td>
</tr>
</tbody>
</table>
Related Information

M_HEAP_MEMORY System View [page 1924]
M_CONTEXT_MEMORY System View [page 1801]
M_SERVICE_MEMORY System View [page 2113]
Memory Management

6.2.129 M_HEAP_MEMORY_RESET System View

Provides memory allocator statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_HEAP_MEMORY. Refer to M_HEAP_MEMORY for information about the structure and use of this view.

In addition to the members mentioned in M_HEAP_MEMORY, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_HEAP_MEMORY System View [page 1924]

6.2.130 M_HISTORY_INDEX_LAST_COMMIT_ID System View

Provides the last commit ID of a history index for each session.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SESSION_ID</td>
<td>BIGINT</td>
<td>Displays the session ID.</td>
</tr>
<tr>
<td>SESSION_OWNER_VOLUME_ID</td>
<td>BIGINT</td>
<td>Displays the volume ID of this session.</td>
</tr>
<tr>
<td>LAST_COMMIT_ID</td>
<td>BIGINT</td>
<td>Displays the last commit ID of history index in this session.</td>
</tr>
</tbody>
</table>
Related Information

COMMIT Statement (Transaction Management) [page 720]
HISTORY COLUMN Option (Time Travel) [page 875]
SAP HANA History Tables

6.2.131  M_HOST_AGENT_INFORMATION System View

Displays computer system information retrieved from the SAP Host Agent.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>CLASS_NAME</td>
<td>NVARCHAR(64)</td>
<td>Displays the name of the class.</td>
</tr>
<tr>
<td>INSTANCE_NUMBER</td>
<td>INTEGER</td>
<td>Displays the number of instances per class, for example, if the host has multiple CPUs, then all the properties of one CPU have the same instance number.</td>
</tr>
<tr>
<td>PROPERTY_NAME</td>
<td>NVARCHAR(64)</td>
<td>Displays the name of the property.</td>
</tr>
<tr>
<td>PROPERTY_VALUE</td>
<td>NVARCHAR(512)</td>
<td>Displays the value of the property.</td>
</tr>
</tbody>
</table>

Related Information

Add Hosts Using SAP Host Agent
Using SAP Host Agent to Execute Platform LCM Tasks
Using SAP Host Agent to Execute Platform LCM Tasks
M_HOST_AGENT_METRICS System View [page 1929]
AGENTS System View [page 1488]
AGENT_CONFIGURATION System View [page 1489]
M_AGENTS System View [page 1744]
## 6.2.132 M_HOST_AGENT_METRICS System View

Displays performance metrics retrieved from the SAP Host Agent.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>CLASS_NAME</td>
<td>NVARCHAR(64)</td>
<td>Displays the name of the class, which identifies the property as a CIM metric or a SAPOSCOL metric.</td>
</tr>
<tr>
<td>INSTANCE_ID</td>
<td>NVARCHAR(256)</td>
<td>Identifies the metric of a specific measured element.</td>
</tr>
<tr>
<td>DEFINITION_ID</td>
<td>NVARCHAR(64)</td>
<td>Displays the ID of the metric.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp from the host agent when the metric instance was created.</td>
</tr>
<tr>
<td>CAPTION</td>
<td>NVARCHAR(64)</td>
<td>Displays the name of the metric.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(128)</td>
<td>Displays the value of the metric.</td>
</tr>
<tr>
<td>UNIT</td>
<td>NVARCHAR(32)</td>
<td>Displays the unit of the metric (bytes, percent, and so on).</td>
</tr>
<tr>
<td>CIM_DATA_TYPE</td>
<td>SMALLINT</td>
<td>Displays a numerical value that identifies the data type of the metric as defined by the CIM standard.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>NVARCHAR(512)</td>
<td>Displays a detailed description of the metric.</td>
</tr>
<tr>
<td>MEASURED_ELEMENT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the measured element: CPU ID, file system name, network port name, or disk name.</td>
</tr>
<tr>
<td>MEASURED_ELEMENT_TYPE</td>
<td>NVARCHAR(64)</td>
<td>Displays the type of measured element (file system, operating system, and so on).</td>
</tr>
</tbody>
</table>
Related Information

M_HOST_AGENT_INFORMATION System View [page 1928]
Add Hosts Using SAP Host Agent
Using SAP Host Agent to Execute Platform LCM Tasks
Using SAP Host Agent to Execute Platform LCM Tasks
AGENTS System View [page 1488]
AGENT_CONFIGURATION System View [page 1489]
M_AGENTS System View [page 1744]

6.2.133 M_HOST_INFORMATION System View

Provides host information such as machine and OS configuration.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>NVARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>KEY</td>
<td>NVARCHAR(64)</td>
<td>Displays the key.</td>
</tr>
</tbody>
</table>

Additional Information

This view contains some host-specific information. The following table describes the contents of the individual lines of the monitor view (VALUE), as identified by KEY. The values are updated once per minute. Most values require the INIFILE ADMIN privilege to view, as noted in the table.

<table>
<thead>
<tr>
<th>KEY</th>
<th>Description</th>
<th>Requires INIFILE ADMIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>Displays the state of the host’s nameserver: Yes (started) or No (stopped).</td>
<td>Y</td>
</tr>
<tr>
<td>async_io_limit</td>
<td>Displays the maximum number of asynchronous input and output requests. The Kernel parameter is fs.aio-max-nr.</td>
<td>Y</td>
</tr>
<tr>
<td>KEY</td>
<td>Description</td>
<td>Requires INFILE ADMIN</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>build_branch</td>
<td>Displays the build branch name in the revision control software.</td>
<td>N</td>
</tr>
<tr>
<td>build_cloud_edition</td>
<td>Displays the Cloud version number for Cloud releases.</td>
<td>Y</td>
</tr>
<tr>
<td>build_compiler</td>
<td>Displays the build compiler.</td>
<td>Y</td>
</tr>
<tr>
<td>build_gen</td>
<td>Displays either &quot;opt&quot; or &quot;dbg&quot;.</td>
<td>Y</td>
</tr>
<tr>
<td>build_time</td>
<td>Displays the build compile time.</td>
<td>Y</td>
</tr>
<tr>
<td>build_version</td>
<td>Displays the build version number. This is the same as M_DATABASE.VERSION.</td>
<td>N</td>
</tr>
<tr>
<td>clocksource.current_clocksource</td>
<td>Displays the clock source.</td>
<td>Y</td>
</tr>
<tr>
<td>container</td>
<td>Displays whether SAP HANA is running inside a Docker container.</td>
<td>Y</td>
</tr>
<tr>
<td>cpu_clock</td>
<td>Displays the CPU clock in MHz.</td>
<td>Y</td>
</tr>
<tr>
<td>cpu_cores - Deprecated</td>
<td>Displays the number of CPU cores.</td>
<td>Y</td>
</tr>
<tr>
<td>cpu_flags</td>
<td>Displays the CPU flags.</td>
<td>Y</td>
</tr>
<tr>
<td>cpu_manufacturer</td>
<td>Displays the CPU manufacturer.</td>
<td>Y</td>
</tr>
<tr>
<td>cpu_model</td>
<td>Displays the CPU model.</td>
<td>Y</td>
</tr>
<tr>
<td>cpu.Sockets</td>
<td>Displays the number of CPU sockets per host.</td>
<td>Y</td>
</tr>
<tr>
<td>cpu_summary</td>
<td>Displays the CPU summary. This is preformatted cpu_cores + cpu_threads + cpu_clock string.</td>
<td>Y</td>
</tr>
<tr>
<td>cpu_threads</td>
<td>Displays the number of logical CPU cores, for example hyper threading.</td>
<td>Y</td>
</tr>
<tr>
<td>cpu.cpubus.freq.scaling_govenor</td>
<td>Displays the cpu_threads mode.</td>
<td>Y</td>
</tr>
<tr>
<td>cpu.cpubusidle.current_driver</td>
<td>Displays the active cpuidle driver.</td>
<td>Y</td>
</tr>
<tr>
<td><strong>KEY</strong></td>
<td><strong>Description</strong></td>
<td><strong>Requires INIFILE ADMIN</strong></td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>cpu.cpuidle.intel_idle.max_cstate</td>
<td>Displays the maximum depth of the C-state. A value of 0 disables intel_idle and the CPU falls back to acpi_idle.</td>
<td>Y</td>
</tr>
<tr>
<td>crypto_fips_provider</td>
<td>Displays whether SAP HANA is using a FIPS 140-2 certified crypto provider for crypto operations.</td>
<td>Y</td>
</tr>
<tr>
<td>crypto_provider</td>
<td>Displays the name of the Cryptographic Provider that is used for SSL, SAML, and so on.</td>
<td>Y</td>
</tr>
<tr>
<td>crypto_provider_version</td>
<td>If SAP HANA is using a FIPS 140-2 certified crypto provider then this column displays the version.</td>
<td>Y</td>
</tr>
<tr>
<td>daemon_active</td>
<td>Displays YES if hdbdaemon is running.</td>
<td>Y</td>
</tr>
<tr>
<td>hw_manufacturer</td>
<td>Displays the hardware manufacturer from BIOS.</td>
<td>Y</td>
</tr>
<tr>
<td>hw_model</td>
<td>Displays the hardware model from BIOS.</td>
<td>Y</td>
</tr>
<tr>
<td>kernel.numa_balancing</td>
<td>Displays 0 if the automatic NUMA (non-uniform memory access) balancing is off.</td>
<td>Y</td>
</tr>
<tr>
<td>kernel.shmall</td>
<td>Displays how much, in 4KB blocks, SysV shared memory is available in the Linux system as a whole.</td>
<td>Y</td>
</tr>
<tr>
<td>kernel.shmmmax</td>
<td>Displays the allowed individual SysV shared memory segment size, in bytes.</td>
<td>Y</td>
</tr>
<tr>
<td>kernel.shmmnmi</td>
<td>Displays the system-wide maximum number of shared memory segments.</td>
<td>Y</td>
</tr>
<tr>
<td>kernel.transparent_hugepage.enabled</td>
<td>Displays the status of the transparent hugepage.</td>
<td>Y</td>
</tr>
<tr>
<td>mem_phys</td>
<td>Displays the physical memory size in bytes.</td>
<td>Y</td>
</tr>
<tr>
<td>mem_swap</td>
<td>Displays the swap size in bytes.</td>
<td>Y</td>
</tr>
<tr>
<td>memory_map_area_limit</td>
<td>Displays the maximum number of virtual memory map areas that</td>
<td>Y</td>
</tr>
<tr>
<td>KEY</td>
<td>Description</td>
<td>Requires INFILE ADMIN</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>net_domain</td>
<td>Displays the domain name of the current host or &quot;-&quot; if not configured.</td>
<td>Y</td>
</tr>
<tr>
<td>net_hostnames</td>
<td>Displays all known host names of this host.</td>
<td>Y</td>
</tr>
<tr>
<td>net_nameserverBindings</td>
<td>Displays all interfaces that the nameserver listens on, as configured with global.ini/ [communication]/listeninterface, [internal_hostname_resolution] and [system_replication...].</td>
<td>Y</td>
</tr>
<tr>
<td>net_publicports</td>
<td>Displays the net public ports.</td>
<td>Y</td>
</tr>
<tr>
<td>net_ip_addresses</td>
<td>Displays all IP addresses of this host.</td>
<td>Y</td>
</tr>
<tr>
<td>net_port_ephemeral_max_count</td>
<td>Displays the number of unreserved ports, which are in net.ipv4.ip_local_port_range.</td>
<td>Y, available only in SYSTEMDB.</td>
</tr>
<tr>
<td>net_port_ranges</td>
<td>Displays the relevant ports for SAP HANA and is dependent on the running services.</td>
<td>Y, available only in SYSTEMDB.</td>
</tr>
<tr>
<td>net_port_unreserved_ranges</td>
<td>Displays the relevant ports for SAP HANA that are not reserved and that are within the net.ipv4.ip_local_port_range.</td>
<td>Y, available only in SYSTEMDB.</td>
</tr>
<tr>
<td>net_publicname</td>
<td>Displays the public host name used by client interfaces. This can contain a host name, FQDN, or IP address.</td>
<td>N</td>
</tr>
<tr>
<td>net_realhostname</td>
<td>Real host name if installation was done with a virtual host name, else set to &quot;.&quot;</td>
<td>Y</td>
</tr>
<tr>
<td>net.core.somaxconn</td>
<td>Displays the size limit of the accept backlog of a listening socket in bytes.</td>
<td>Y</td>
</tr>
<tr>
<td>net.ipv4.tcp_max_syn_backlog</td>
<td>Displays the size of the SYN backlog in bytes.</td>
<td>Y</td>
</tr>
<tr>
<td>net.ipv4.tcp_rmem</td>
<td>Displays the minimum, default, and maximum size of the TCP receive buffer in bytes.</td>
<td>Y</td>
</tr>
<tr>
<td>KEY</td>
<td>Description</td>
<td>Requires INFILE ADMIN</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>net.ipv4.tcp_syn_retries</td>
<td>Displays the number of times initial SYNs for an active TCP connection attempt are retransmitted.</td>
<td>Y</td>
</tr>
<tr>
<td>net.ipv4.tcp_timestamps</td>
<td>Displaying 1 enables TCP timestamps, adding the timestamp field to the TCP header.</td>
<td>Y</td>
</tr>
<tr>
<td>net.ipv4.tcp_tw_recycle</td>
<td>Displaying 1 reduces the time a connection spends in the TIME_WAIT state. The precondition is net.ipv4.tcp_timestamps = 1.</td>
<td>Y</td>
</tr>
<tr>
<td>net.ipv4.tcp_tw_reuse</td>
<td>Displaying 1 allows SAP HANA to reuse a client port immediately after the connection is closed. The precondition is net.ipv4.tcp_timestamps = 1.</td>
<td>Y</td>
</tr>
<tr>
<td>net.ipv4.tcp_window_scaling</td>
<td>Displays if the TCP window scaling is enabled: TRUE/FALSE.</td>
<td>Y</td>
</tr>
<tr>
<td>net.ipv4.tcp_wmem</td>
<td>Displays the minimum, default, and maximum size of the TCP send buffer in bytes.</td>
<td>Y</td>
</tr>
<tr>
<td>open_file_limit</td>
<td>Displays the maximum number of open file descriptor handles on the host. The Kernel parameter is fs.file-max. For more information, see the OPEN_FILE_COUNT value in the M_HOST_RESOURCE_UTILIZATION System View topic.</td>
<td>Y</td>
</tr>
<tr>
<td>os_cpe_name</td>
<td>Displays operating system information.</td>
<td>Y</td>
</tr>
<tr>
<td>os_kernel_version</td>
<td>Displays the Linux kernel version.</td>
<td>Y</td>
</tr>
<tr>
<td>os_name</td>
<td>Displays the operating system name, as shown in the PRETTY_NAME field in /etc/os-release.</td>
<td>Y</td>
</tr>
<tr>
<td>os_ppms_name</td>
<td>Displays the operating system name as used by SAP PPMS.</td>
<td>Y</td>
</tr>
<tr>
<td>os_rlimit_nofile</td>
<td>Displays the Linux RLIMIT_NOFILE value.</td>
<td>Y</td>
</tr>
<tr>
<td>KEY</td>
<td>Description</td>
<td>Requires INFILE ADMIN</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>os_user</td>
<td>Displays the operating system user.</td>
<td>Y</td>
</tr>
<tr>
<td>platform</td>
<td>Displays &quot;windows&quot; or &quot;unix&quot;.</td>
<td>Y</td>
</tr>
<tr>
<td>sap_retrieval_path</td>
<td>Displays the host-specific base directory.</td>
<td>Y</td>
</tr>
<tr>
<td>sap_retrieval_path_shared</td>
<td>Displays &quot;yes&quot; if all sap_retrieval_path directories are located on a shared storage.</td>
<td>Y</td>
</tr>
<tr>
<td>sapsystem</td>
<td>Displays the two digit SAP system instance number.</td>
<td>N</td>
</tr>
<tr>
<td>secure_store</td>
<td>Displays &quot;available&quot; if the secure store is present and readable.</td>
<td>Y</td>
</tr>
<tr>
<td>sid</td>
<td>Displays the three letter SAP system name.</td>
<td>N</td>
</tr>
<tr>
<td>start_time</td>
<td>Displays the start time of the name server.</td>
<td>Y</td>
</tr>
<tr>
<td>timezone_base_name</td>
<td>Displays the base time zone name.</td>
<td>Y</td>
</tr>
<tr>
<td>timezone_dst_name</td>
<td>Displays the daylight savings time zone name.</td>
<td>Y</td>
</tr>
<tr>
<td>timezone_name</td>
<td>Displays the time zone name.</td>
<td>Y</td>
</tr>
<tr>
<td>timezone_offset</td>
<td>Displays the time zone offset to GMT, in seconds.</td>
<td>Y</td>
</tr>
<tr>
<td>timezone_tz_name</td>
<td>Displays the time zone set in the environment variable TZ.</td>
<td>Y</td>
</tr>
<tr>
<td>topology_mem_info</td>
<td>Displays an error message if shared memory could not be created.</td>
<td>Y</td>
</tr>
<tr>
<td>topology_mem_type</td>
<td>Always specified as &quot;shared&quot;.</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Related Information**

M_HOSTRESOURCE_UTILIZATION System View [page 1937]
M_HOST_NETWORK_STATISTICS System View [page 1936]
Configurations and Configuration Templates
Default Host Names and Virtual Host Names
Add Hosts Using the Command-Line Interface
Use FIPS 140-2 Certified Cryptographic Kernel in CommonCryptoLib
https://launchpad.support.sap.com/#/notes/2382421
https://www.kernel.org/doc/Documentation/sysctl/fs.txt
https://www.kernel.org/doc/Documentation/sysctl/vm.txt
How to configure HANA network communication channels – Part1. Public network

6.2.134 M_HOST_NETWORK_STATISTICS System View

Provides information about the network statistics of a host.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name the network statistics refer to.</td>
</tr>
<tr>
<td>TCP_SEGMENTS_RECEIVED</td>
<td>BIGINT</td>
<td>Displays the total number of TCP segments received.</td>
</tr>
<tr>
<td>TCP_SEGMENTS_SENT_OUT</td>
<td>BIGINT</td>
<td>Displays the total number of TCP segments sent out.</td>
</tr>
<tr>
<td>TCP_SEGMENTS_RETRANSMITTED</td>
<td>BIGINT</td>
<td>Displays the number of TCP segments that had to be retransmitted.</td>
</tr>
<tr>
<td>TCP_BAD_SEGMENTS_RECEIVED</td>
<td>BIGINT</td>
<td>Displays the number of TCP segments that were broken upon receiving.</td>
</tr>
</tbody>
</table>

Additional Information

Most columns correspond to the similarly named values reported by `netstat -s`.

In a SystemDB the view returns one row for each physical host system present. In a tenant database, the view returns one row for each physical host system on which at least one service process of the respective tenant database is running.
Related Information

M_HOST_INFORMATION System View [page 1930]
M_HOST_RESOURCE_UTILIZATION System View [page 1937]
Configuring the Network for Multiple Hosts
Monitoring the Network Between Multiple Hosts

6.2.135 M_HOST_RESOURCE_UTILIZATION System View

Provides information about host resource utilization by all processes (including non-SAP HANA processes). CPU time is in milliseconds and added across all cores since system start.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>FREE_PHYSICAL_MEMORY</td>
<td>BIGINT</td>
<td>Displays the free physical memory on the host in bytes.</td>
</tr>
<tr>
<td>USED_PHYSICAL_MEMORY</td>
<td>BIGINT</td>
<td>Displays the used physical memory on the host in bytes.</td>
</tr>
<tr>
<td>FREE_SWAP_SPACE</td>
<td>BIGINT</td>
<td>Displays the free swap memory on the host in bytes.</td>
</tr>
<tr>
<td>USED_SWAP_SPACE</td>
<td>BIGINT</td>
<td>Displays the used swap memory on the host in bytes.</td>
</tr>
<tr>
<td>ALLOCATION_LIMIT</td>
<td>BIGINT</td>
<td>Displays the allocation limit for all processes in bytes.</td>
</tr>
<tr>
<td>INSTANCE_TOTAL_MEMORY_USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the amount of memory from the memory pool that is currently being used by SAP HANA processes in bytes.</td>
</tr>
<tr>
<td>INSTANCE_TOTAL_MEMORY_PEAK_USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the peak memory from the memory pool used by SAP HANA processes since the instance started (this is a sample-based value) in bytes.</td>
</tr>
<tr>
<td>INSTANCE_TOTAL_MEMORY_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the memory pool for all SAP HANA processes in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INSTANCE_CODE_SIZE</td>
<td>BIGINT</td>
<td>Displays the code size, including shared libraries of SAP HANA processes in bytes.</td>
</tr>
<tr>
<td>INSTANCE_SHARED_MEMORY_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the shared memory size of SAP HANA processes in bytes.</td>
</tr>
<tr>
<td>TOTAL_CPU_USER_TIME</td>
<td>BIGINT</td>
<td>Displays the CPU time spent in the user mode in milliseconds.</td>
</tr>
<tr>
<td>TOTAL_CPU_SYSTEM_TIME</td>
<td>BIGINT</td>
<td>Displays the CPU time spent in the kernel mode in milliseconds.</td>
</tr>
<tr>
<td>TOTAL_CPU_WIO_TIME</td>
<td>BIGINT</td>
<td>Displays the CPU time spent in wait I/O in milliseconds. Linux only. Microsoft Windows is always 0.</td>
</tr>
<tr>
<td>TOTAL_CPU_IDLE_TIME</td>
<td>BIGINT</td>
<td>Displays the CPU idle time in milliseconds.</td>
</tr>
<tr>
<td>SYS_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the host timestamp in the local time zone.</td>
</tr>
<tr>
<td>UTC_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the host timestamp in UTC.</td>
</tr>
<tr>
<td>OPEN_FILE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of allocated file handles on the host. The Kernel parameter is $\text{fs.file-nr}$. For more information, see the OPEN_FILE_LIMIT key in the M_HOST_INFORMATION system view topic.</td>
</tr>
<tr>
<td>ACTIVE_ASYNC_IO_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of asynchronous input and/or output requests on the host. The Kernel parameter is $\text{fs.ai-nr}$. For more information, see the ASYNC_IO_LIMIT key in the M_HOST_INFORMATION system view topic.</td>
</tr>
</tbody>
</table>

**Related Information**

HOST_RESOURCE_UTILIZATION_STATISTICS View (Embedded Statistics Service)
M_HOST_INFORMATION System View [page 1930]
M_HOST_NETWORK_STATISTICS System View [page 1936]
View Utilized System Resources
Add or Remove Databases in a Database Group

6.2.136  M_IMPORT_BINARY_STATUS System View

Provides import status information for the current session.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID to distinguish data of each import session (-1 for Python).</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema of the table or view being imported.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the table or view being imported.</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the index type: UNKNOWN, PHYSICAL, OLAP, JOIN, HIERARCHY, or CALCULATION.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the import status: QUEUED, WORKING, SKIPPED, DONE, or FAILED.</td>
</tr>
<tr>
<td>ERROR</td>
<td>VARCHAR(512)</td>
<td>Displays empty if the import was successful and displays error text otherwise.</td>
</tr>
</tbody>
</table>

Related Information

Binary Data Types [page 35]
IMPORT Statement (Data Import Export) [page 1029]
IMPORT FROM Statement (Data Import Export) [page 1038]
### 6.2.137 M_INDEXING_QUEUES System View

Provides the status of indexing queues.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>QUEUE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of the queue: TEXT/GEOCODE.</td>
</tr>
<tr>
<td>INDEX_OID</td>
<td>BIGINT</td>
<td>Displays the OID of the corresponding index.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema of the data table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the data table.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is in progress.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the document column.</td>
</tr>
<tr>
<td>STATUS</td>
<td>NVARCHAR(16)</td>
<td>Displays the queue status: ACTIVE/SUSPENDED.</td>
</tr>
<tr>
<td>TOTAL_DOCUMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of documents in the original table.</td>
</tr>
<tr>
<td>INDEXED_DOCUMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of successfully indexed documents.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>QUEUE_DOCUMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of documents in the queue.</td>
</tr>
<tr>
<td>ERROR_DOCUMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of failed documents.</td>
</tr>
</tbody>
</table>

**Related Information**

INDEXING_STATUS Function (Fulltext) [page 216]
INDEXES System View [page 1585]
ALTER INDEX Statement (Data Definition) [page 452]
CREATE INDEX Statement (Data Definition) [page 762]
RENAME INDEX Statement (Data Definition) [page 1091]
DROP INDEX Statement (Data Definition) [page 958]

**6.2.138  M_INIFILES System View**

Provides information about all configuration files.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the configuration file name.</td>
</tr>
<tr>
<td>DEFAULT_LAYER</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the file has configuration on the default layer. This value is always TRUE.</td>
</tr>
<tr>
<td>SYSTEM_LAYER</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the file has configuration on the system layer: TRUE/FALSE.</td>
</tr>
<tr>
<td>DATABASE_LAYER</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the file has configuration on the database layer: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
HOST_LAYER | VARCHAR(5) | Indicates whether or not the file has configuration on the host layer: TRUE/ FALSE.

### Additional Information

You can update configuration file settings using the ALTER SYSTEM ALTER CONFIGURATION statement.

### Related Information

- Configuring SAP HANA System Properties (INI Files)
- M_INIFILE_CONTENTS System View [page 1942]
- M_INIFILE_CONTENT_HISTORY System View [page 1943]
- ALTER SYSTEM ALTER CONFIGURATION Statement (System Management) [page 500]
- ALTER SYSTEM ALTER CONFIGURATION Statement [Dynamic Tiering] [page 1312]

### 6.2.139 M_INIFILE_CONTENTS System View

Provides configuration information from INI files.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the configuration file name.</td>
</tr>
<tr>
<td>LAYER_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the configuration layer: DEFAULT, SYSTEM, HOST or DATABASE.</td>
</tr>
<tr>
<td>TENANT_NAME</td>
<td>VARCHAR(256)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name, if the LAYER_NAME is HOST. Otherwise, this value is NULL.</td>
</tr>
</tbody>
</table>
### Related Information

Configuring SAP HANA System Properties (INI Files)
- M_INIFILES System View [page 1941]
- M_INIFILE_CONTENT_HISTORY System View [page 1943]
- ALTER SYSTEM ALTER CONFIGURATION Statement (System Management) [page 500]

### 6.2.140 M_INIFILE_CONTENT_HISTORY System View

Provides change history information for configuration (ini) files.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the configuration setting was changed.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the name of the configuration file.</td>
</tr>
<tr>
<td>LAYER_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the configuration layer: DEFAULT, SYSTEM, HOST, or DATABASE.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name. If the LAYER_NAME is HOST. Otherwise, this value is NULL.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name if the change was made by an SAP HANA user.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the value of the APPLICATION session variable, if an application made the change. Otherwise, this value is NULL.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the value of the APPLICATION_USER_NAME session variable, if an application made the change. Otherwise, this value is NULL.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application source if an application made the change. Otherwise, this value is NULL.</td>
</tr>
<tr>
<td>SECTION</td>
<td>VARCHAR(128)</td>
<td>Displays the name of the section in the configuration file where the change was made.</td>
</tr>
<tr>
<td>KEY</td>
<td>VARCHAR(128)</td>
<td>Displays the name of the configuration key that was changed.</td>
</tr>
<tr>
<td>VALUE</td>
<td>VARCHAR(5000)</td>
<td>Displays the new configuration value.</td>
</tr>
<tr>
<td>PREV_VALUE</td>
<td>VARCHAR(5000)</td>
<td>Displays the value of the configuration parameter before the change was made.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the comment provided, if any, at the time of the configuration change.</td>
</tr>
</tbody>
</table>

**Related Information**

- ALTER SYSTEM CLEAR INIFILE CONTENT HISTORY Statement (System Management) [page 521]
- Configuring SAP HANA System Properties (INI Files)
- M_INIFILES System View [page 1941]
- M_INIFILE_CONTENTS System View [page 1942]
- ALTER SYSTEM ALTER CONFIGURATION Statement (System Management) [page 500]
### 6.2.141 M_JOBEXECUTORS System View

Provides job executor statistics.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>MAX_CONCURRENCY</td>
<td>INTEGER</td>
<td>Displays the maximum number of concurrently running worker threads. The job executor keeps MAX_CONCURRENCY worker threads busy. As a result, more worker threads are started if some worker threads are in a wait state for some time. Even if these additional worker threads are terminated, some of them are kept for later use and counted in PARKED_WORKER_COUNT.</td>
</tr>
<tr>
<td>TOTAL_WORKER_COUNT</td>
<td>INTEGER</td>
<td>Displays the total number of worker threads, including one extra worker thread for emergency diagnostic purposes.</td>
</tr>
<tr>
<td>PARKED_WORKER_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of parked worker threads.</td>
</tr>
<tr>
<td>FREE_WORKER_COUNT</td>
<td>INTEGER</td>
<td>Displays the total number of worker threads that are idle and may immediately take up a job to work on.</td>
</tr>
<tr>
<td>SYS_WAITING_WORKER_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of worker threads waiting in system. A worker thread may wait for any other kind of synchronization.</td>
</tr>
<tr>
<td>JOB_WAITING_WORKER_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of worker threads waiting for a job on another worker thread.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>YIELD_WAITING_WORKER_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of worker threads waiting on a yield. Sometimes a job running on a worker thread yields execution to another job that is more important.</td>
</tr>
<tr>
<td>TOTAL_WAITING_JOB_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of all jobs waiting for execution.</td>
</tr>
<tr>
<td>QUEUED_WAITING_JOB_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of jobs queued for execution.</td>
</tr>
<tr>
<td>WORKER_CREATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of threads created.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view has a resettable counterpart; you can see the values since the last reset in the M_JOBEXECUTORS_RESET system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_JOBEXECUTORS_RESET;`.

**Related Information**

- M_JOBEXECUTORS_RESET System View [page 1946]
- Controlling Parallel Execution of SQL Statements
- Scheduling Jobs in XS Advanced
- Maintain Jobs and Job Schedules in XS Advanced
- The Job Scheduler Dashboard

**6.2.142 M_JOBEXECUTORS_RESET System View**

Provides values accumulated since the last reset of the main view M_JOBEXECUTORS.

In addition to the members mentioned in M_JOBEXECUTORS, this view also contains a timestamp field, `RESET_TIME`, which indicates the last time the data was reset.

Refer to M_JOBEXECUTORS for information about the structure and use of this view.
Related Information

M_JOBEXECUTORS System View [page 1945]

6.2.143 M_JOB_HISTORY_INFO System View

Provides a history of long running system operations.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB_NAME</td>
<td>NVARCHAR(128)</td>
<td>Displays the type of operation.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(256)</td>
<td>Displays the job description.</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the recommended display name.</td>
</tr>
<tr>
<td>DISPLAY_LINE_COLOR</td>
<td>INTEGER(10)</td>
<td>Displays the recommended display line color as RGB.</td>
</tr>
<tr>
<td>DISPLAY_LINE_STYLE</td>
<td>TINYINT(3)</td>
<td>Displays the recommended display line style: 1=solid, 2=dotted, 3=dashed.</td>
</tr>
</tbody>
</table>

Related Information

M_JOB_PROGESS System View [page 1948]
SAP HANA History Tables
The Job Scheduler Dashboard
View Change History
6.2.144 M_JOB_PROGRESS System View

Provides information about current long running system operations.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema of the object.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object.</td>
</tr>
<tr>
<td>JOB_NAME</td>
<td>NVARCHAR(128)</td>
<td>Displays the type of operation. See the table below for more detail.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection that triggered the operation.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time of the operation.</td>
</tr>
<tr>
<td>CURRENT_PROGRESS</td>
<td>INTEGER</td>
<td>Displays the steps of the operation that are already finished.</td>
</tr>
<tr>
<td>MAX_PROGRESS</td>
<td>INTEGER</td>
<td>Displays the maximum operation progress.</td>
</tr>
<tr>
<td>PROGRESS_DETAIL</td>
<td>NVARCHAR(256)</td>
<td>Displays the detailed information about the operation.</td>
</tr>
</tbody>
</table>
## Additional Information

The below table lists the possible types of job operations:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB_NAME</td>
<td>• Alter Column Load Unit: Convert/change load unit at the column level using NSE DDLs.</td>
</tr>
<tr>
<td></td>
<td>• Alter Index Load Unit: Convert/change load unit at the index level using NSE DDLs.</td>
</tr>
<tr>
<td></td>
<td>• Alter Partition Load Unit: Convert/change load unit at the partition level using NSE DDLs.</td>
</tr>
<tr>
<td></td>
<td>• Alter Table Load Unit: Convert/change load unit at the table level using NSE DDLs.</td>
</tr>
<tr>
<td></td>
<td>• BW F Fact Table Compression: fact table compression</td>
</tr>
<tr>
<td></td>
<td>• Backup: data backup</td>
</tr>
<tr>
<td></td>
<td>• Check Table Consistency: table consistency check</td>
</tr>
<tr>
<td></td>
<td>• Column table reloading on startup: Column table reloading on startup</td>
</tr>
<tr>
<td></td>
<td>• Create Index: index creation</td>
</tr>
<tr>
<td></td>
<td>• DSO activation: dso activation</td>
</tr>
<tr>
<td></td>
<td>• DSO conversion: dso conversion</td>
</tr>
<tr>
<td></td>
<td>• DSO rollback: dso rollback</td>
</tr>
<tr>
<td></td>
<td>• Data Statistics Autocreate: Automatic creation of data statistics</td>
</tr>
<tr>
<td></td>
<td>• Delta Log Replay: delta log replay</td>
</tr>
<tr>
<td></td>
<td>• Delta Merge: delta merge</td>
</tr>
<tr>
<td></td>
<td>• Export All: main export operation</td>
</tr>
<tr>
<td></td>
<td>• Export Object: exporting data for an object invoked by ‘Export All’</td>
</tr>
<tr>
<td></td>
<td>• Import All: main import operation</td>
</tr>
<tr>
<td></td>
<td>• Import Object: importing data for an object invoked by ‘Import All’</td>
</tr>
<tr>
<td></td>
<td>• Memory Profiler: load memory profiler data into tables</td>
</tr>
<tr>
<td></td>
<td>• Move Table: table move</td>
</tr>
<tr>
<td></td>
<td>• Online Repartitioning: allows DML workloads during DDL executions</td>
</tr>
<tr>
<td></td>
<td>• Optimize Compression: optimize compression</td>
</tr>
<tr>
<td></td>
<td>• Plan Stability: Execute all cached plans to capture</td>
</tr>
<tr>
<td></td>
<td>• Re-partitioning: table split and merge operations</td>
</tr>
<tr>
<td></td>
<td>• Reclaim Delta: reclaim rows from delta which are not visible</td>
</tr>
<tr>
<td></td>
<td>• Row Store Reorganization: row storage reorganization</td>
</tr>
<tr>
<td></td>
<td>• Runtimedump: Runtimedump</td>
</tr>
</tbody>
</table>
Related Information

Job Progress Monitoring
M_JOB_HISTORY_INFO System View [page 1947]

6.2.145 M_JOINENGINE_STATISTICS System View

Provides statistics about join engine runtime objects used for column store join operations.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>TRANSLATION_TABLE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of translation tables.</td>
</tr>
</tbody>
</table>
### Related Information

- **M_JOIN_DATA_STATISTICS System View [page 1951]**
- **M_JOIN_TRANSLATION_TABLES System View [page 1954]**
- **M_TABLE_PRUNING_STATISTICS System View [page 2218]**

### Process Engine Roles

**6.2.146 M_JOIN_DATA_STATISTICS System View**

Provides column store join engine join statistics.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the first table</td>
</tr>
<tr>
<td>TABLE_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the first table</td>
</tr>
<tr>
<td>COLUMN_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name of the first table</td>
</tr>
<tr>
<td>SCHEMA_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the second table. Join data statistics are not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>collected</td>
</tr>
<tr>
<td>TABLE_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the second table.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COLUMN_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name of the second table.</td>
</tr>
<tr>
<td>LAST_REFRESH_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time when the statistics was last refreshed.</td>
</tr>
<tr>
<td>RECORD_COUNT1</td>
<td>BIGINT</td>
<td>Displays the estimated total row count of the first table. For temporary tables, this value is always 0.</td>
</tr>
<tr>
<td>COUNT1</td>
<td>BIGINT</td>
<td>Displays the number of not Null values of the column of the first table.</td>
</tr>
<tr>
<td>DISTINCT_COUNT1</td>
<td>BIGINT</td>
<td>Displays the number of unique not Null values of the column of the first table.</td>
</tr>
<tr>
<td>VALUE_COUNTS1</td>
<td>NVARCHAR(1000)</td>
<td>Displays a comma-separated sorted list of unique column value counts. An incomplete output of the list is truncated by an asterisk sign. If a uniform distribution of column values is assumed, then there is exactly one value count because values are uniformly distributed and every value counts the same.</td>
</tr>
<tr>
<td>VALUE_COUNTS_FREQUENCIES1</td>
<td>NVARCHAR(1000)</td>
<td>Displays a comma-separated list of frequencies of the first table’s column values with the same count. Any frequency refers to that value count with a corresponding position in the list of value counts. An incomplete output of the list is truncated by an asterisk sign. If a uniform distribution of column values is assumed, then there is exactly one value frequency because values are uniformly distributed. Generally, this single frequency matches the distinct count.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MATCHING_VALUES_COUNTS1</td>
<td>NVARCHAR(1000)</td>
<td>Displays a comma-separated list of counts of the first table’s column values which find a join partner. Any matching values count refers to that value count with corresponding position in the list of value counts. An incomplete output of the list is truncated by an asterisk sign. If a uniform distribution of column values is assumed, then the matching values count the one and only value count that is calculated from the unified distribution of the value counts.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size, in bytes, used by the join data statistics object.</td>
</tr>
</tbody>
</table>
6.2.147 M_JOIN_TRANSLATION_TABLES System View

Provides column store join engine translation tables statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the first schema name.</td>
</tr>
<tr>
<td>TABLE_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the first table name.</td>
</tr>
<tr>
<td>COLUMN_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the first table's column name.</td>
</tr>
<tr>
<td>PART_ID1</td>
<td>INTEGER</td>
<td>Displays the first table partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ble and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in progress.</td>
</tr>
<tr>
<td>VERSION1</td>
<td>BIGINT</td>
<td>Displays the version information of the first table.</td>
</tr>
<tr>
<td>SCHEMA_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the second schema name.</td>
</tr>
<tr>
<td>TABLE_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the second table name.</td>
</tr>
<tr>
<td>COLUMN_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the second table's column name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PART_ID2</td>
<td>INTEGER</td>
<td>Displays the second table partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is in progress.</td>
</tr>
<tr>
<td>VERSION2</td>
<td>BIGINT</td>
<td>Displays the version information of the second table.</td>
</tr>
<tr>
<td>IMPLEMENTATION_TYPE</td>
<td>NVARCHAR(8)</td>
<td>Displays the implementation type.</td>
</tr>
<tr>
<td>IMPLEMENTATION_TYPE_CHANGE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time that the implementation type was changed.</td>
</tr>
<tr>
<td>IMPLEMENTATION_TYPE_CHANGE_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of implementation type changes.</td>
</tr>
<tr>
<td>VALUE_ID_COUNT1</td>
<td>BIGINT</td>
<td>Displays the number of translated value IDs of the first table.</td>
</tr>
<tr>
<td>VALUE_ID_COUNT2</td>
<td>BIGINT</td>
<td>Displays the number of translated value IDs of the second table.</td>
</tr>
<tr>
<td>TRANSLATION_TABLE_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory used by the translation table in bytes.</td>
</tr>
<tr>
<td>ESTIMATED_HASHMAP_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the estimated memory used by the HASHMAP implementation in bytes.</td>
</tr>
<tr>
<td>ESTIMATED_VECTOR_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the estimated memory used by the vector implementation in bytes.</td>
</tr>
<tr>
<td>LAST_USED_COUNT</td>
<td>BIGINT</td>
<td>Displays the counter to support LRU cache replacement algorithm.</td>
</tr>
</tbody>
</table>

**Related Information**

M_JOINENGINE_STATISTICS System View [page 1950]
6.2.148 M_KERNEL_PROFILER System View

Displays the state and provides information about Kernel Profilers in the system. You must have the RESOURCE ADMIN or TRACE ADMIN system privileges to use this view.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user name to filter on.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the SQL user name to filter on.</td>
</tr>
<tr>
<td>WAIT_TIME</td>
<td>BIGINT</td>
<td>Deprecated. Displays the wait time between two sampling runs, in milliseconds (SAMPLING_INTERVAL)</td>
</tr>
<tr>
<td>MEMORY_LIMIT</td>
<td>BIGINT</td>
<td>Displays the limit for profiling data in bytes. This is not an exact limit, a heuristic approach is used to limit memory.</td>
</tr>
<tr>
<td>SAMPLING_INTERVAL</td>
<td>BIGINT</td>
<td>Displays the sampling interval, in milliseconds.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID to filter on.</td>
</tr>
<tr>
<td>ROOT_STATEMENT_HASH</td>
<td>NVARCHAR(32)</td>
<td>Displays the root statement hash to filter on.</td>
</tr>
<tr>
<td>TRACEPROFILE_NAME</td>
<td>NVARCHAR(128)</td>
<td>Displays the user-specific trace profile name to filter on.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that profiling was started.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
---|---|---
STOP_TIME | TIMESTAMP | Displays the time that profiling was stopped.
MEMORY_SIZE | BIGINT | Displays the current size, in bytes, of the kernel profile.
SAMPLE_COUNT | BIGINT | Displays the number of samples stored for the profile.
STATUS | | Displays the status of the profiler. Possible values are: STARTED, STOPPED

**Related Information**

- ALTER SYSTEM {START | STOP | SAVE | CLEAR} KERNEL PROFILER Statement (System Management) [page 571]
- Kernel Profiler
- Diagnostic Files and Logs
- Database Trace (Basic, User-Specific, and End-to-End)

### 6.2.149 M_LANDSCAPE_HOST_CONFIGURATION System View

Specifies the host roles in a distributed landscape.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>HOST_ACTIVE</td>
<td>VARCHAR(128)</td>
<td>Displays the host active status which is a summary of the active values of all the services on that host.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HOST_STATUS</td>
<td>VARCHAR(128)</td>
<td>Displays the host status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OK</strong> The host is operational; the host actual role equals the configured role.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>IGNORE</strong> The host is operational; the host configured as standby is available, but not used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>INFO</strong> The host is operational; the host actual role is different from the configured role.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>WARNING</strong> The host is not operational; the host should become operational after startup/failover.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ERROR</strong> The host not operational; the host is in an erroneous state.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| FAILOVER_STATUS         | VARCHAR(128)   | Displays the failover status:  
  (empty value)  
  No failover is pending or active.  
  waiting .. sec  
  A failure is detected. Waiting is required to prevent an unnecessary failover. A restart of another host does not trigger an immediate failover.  
  waiting  
  A failure is detected. Waiting for another starting host.  
  failover to 'host'  
  Failover to an active host named 'host'.  
  failback to 'host'  
  Failback to worker host active. This happens when a standby host was assigned and is stopped. There is no automatic failback while the standby host is assigned because this would cause downtime.  
  failed  
  The failover failed. See nameserver.trc for details |
| FAILOVER_GROUP          | VARCHAR(256)   |Deprecated. This has been replaced by the FAILOVER_..._GROUP columns. |
| FAILOVER_CONFIG_GROUP   | VARCHAR(256)   | Displays the configured failover group.  
  Hosts can be grouped and during failover a host is looked for within the same group. If no host is available in the same group, hosts from other groups are used. |
<p>| FAILOVER_ACTUAL_GROUP   | VARCHAR(256)   | Displays the actual failover group. |
| NAMESERVER_CONFIG_ROLE  | VARCHAR(16)    | Displays the nameserver configured roles: MASTER/SLAVE. During installation, up to 3 hosts are automatically configured as MASTER candidates. All others are SLAVE. |</p>
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAMESERVER_ACTUAL_ROLE</td>
<td>VARCHAR(16)</td>
<td>Displays the nameserver actual roles: MASTER/SLAVE. Exactly one of the master candidates is MASTER and all others are SLAVE. The MASTER name server and MASTER index server are both on the same host.</td>
</tr>
<tr>
<td>INDEXSERVER_CONFIG_ROLE</td>
<td>VARCHAR(16)</td>
<td>Displays the indexserver configured roles: WORKER/STANDBY. During SAP HANA system installation or host addition, hosts can be assigned the SAP HANA server roles WORKER or STANDBY. The host can also be assigned other roles.</td>
</tr>
<tr>
<td>INDEXSERVER_ACTUAL_ROLE</td>
<td>VARCHAR(16)</td>
<td>Displays the indexserver roles: MASTER, SLAVE, or STANDBY. Exactly one host is MASTER and all others are SLAVE or STANDBY.</td>
</tr>
<tr>
<td>HOST_CONFIG_ROLES</td>
<td>VARCHAR(64)</td>
<td>Displays the host's configured database and SAP HANA option roles: ETS_WORKER, ETS_STANDBY, EXTENDED_STORAGE_WORKER, EXTENDED_STORAGE_STANDBY, STANDBY, STREAMING, WORKER, XS_WORKER, or XS_STANDBY.</td>
</tr>
<tr>
<td>HOST_ACTUAL_ROLES</td>
<td>VARCHAR(64)</td>
<td>Displays the host's current database and SAP HANA option roles (these may change due to host auto-failover): ETS_WORKER, ETS_STANDBY, EXTENDED_STORAGE_WORKER, EXTENDED_STORAGE_STANDBY, STANDBY, STREAMING, WORKER, XS_WORKER, or XS_STANDBY.</td>
</tr>
<tr>
<td>STORAGE_PARTITION</td>
<td></td>
<td>Deprecated. This has been replaced by the STORAGE_CONFIG_PARTITION, and STORAGE_ACTUAL_PARTITION columns.</td>
</tr>
<tr>
<td>STORAGE_CONFIG_PARTITION</td>
<td>INTEGER</td>
<td>Displays the stable subpath to reassign the same storage partition after failovers.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STORAGE_ACTUAL_PARTITION</td>
<td>INTEGER</td>
<td>Displays the unique number of the mount sub-directory used by the host for storing data and logs (also known as the &quot;subpath&quot;).</td>
</tr>
<tr>
<td>WORKER_CONFIG_GROUPS</td>
<td>VARCHAR(256)</td>
<td>Displays the stable classification values to assign hosts to logical worker groups.</td>
</tr>
<tr>
<td>WORKER_ACTUAL_GROUPS</td>
<td>VARCHAR(256)</td>
<td>Displays the current classification values to assign hosts to logical worker groups.</td>
</tr>
<tr>
<td>REMOVE_STATUS</td>
<td>VARCHAR(16)</td>
<td>Removes progress information.</td>
</tr>
</tbody>
</table>

**Related Information**

Landscape Management  
Multiple-Host System Concepts  
Configuring Clients for Failover  
Monitoring Host Status and Auto-Failover Configuration

**6.2.150 M_LICENSE System View**

Provides information on the currently valid license for the SAP HANA database that is installed on this system.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARDWARE_KEY</td>
<td>VARCHAR(11)</td>
<td>Displays the hardware key of the SAP HANA installation.</td>
</tr>
<tr>
<td>SYSTEM_ID</td>
<td>VARCHAR(3)</td>
<td>Displays the System Identifier (SID).</td>
</tr>
<tr>
<td>INSTALL_NO</td>
<td>VARCHAR(10)</td>
<td>Displays the installation number.</td>
</tr>
<tr>
<td>SYSTEM_NO</td>
<td>VARCHAR(18)</td>
<td>Displays the system number.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PRODUCT_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the name of the licensed software product, for example, SAP HANA.</td>
</tr>
<tr>
<td>PRODUCT_LIMIT</td>
<td>BIGINT</td>
<td>Displays the licensed amount of main memory as specified by the license in bytes.</td>
</tr>
<tr>
<td>PRODUCT_USAGE</td>
<td>BIGINT</td>
<td>Displays the peak memory allocation since installation, measured hourly, in bytes.</td>
</tr>
<tr>
<td>START_DATE</td>
<td>TIMESTAMP</td>
<td>Displays the start date of the validity period of the license.</td>
</tr>
<tr>
<td>EXPIRATION_DATE</td>
<td>TIMESTAMP</td>
<td>Displays the expiration date of the validity period of the license.</td>
</tr>
<tr>
<td>LAST_SUCCESSFUL_CHECK</td>
<td>TIMESTAMP</td>
<td>Displays the latest date on which the license was successfully checked and found valid.</td>
</tr>
<tr>
<td>PERMANENT</td>
<td>VARCHAR(5)</td>
<td>Displays TRUE if the license is permanent and FALSE if the license is temporary.</td>
</tr>
<tr>
<td>VALID</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the license is valid or not: TRUE/FALSE.</td>
</tr>
<tr>
<td>ENFORCED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the license is a memory-enforced license or not: TRUE/ FALSE.</td>
</tr>
<tr>
<td>LOCKED_DOWN</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the system is locked down due to license status: TRUE/ FALSE.</td>
</tr>
<tr>
<td>IS_DATABASE_LOCAL</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the license is a local database local license: TRUE/ FALSE. If FALSE, then the license is provided globally by the SystemDB.</td>
</tr>
<tr>
<td>USAGE</td>
<td>VARCHAR(16)</td>
<td>Displays the database license usage type: NONPRODUCTION, RUNTIME, FULL, or ADMINISTRATION.</td>
</tr>
<tr>
<td>MEASUREMENT_XML</td>
<td>CLOB</td>
<td>Displays the measurement log produced for this licensed system with the current memory consumption.</td>
</tr>
</tbody>
</table>
Related Information

M_LICENSES System View [page 1963]
M_LICENSE_MEASUREMENTS System View [page 1965]
M_LICENSE_MEASUREMENT_STATISTICS System View [page 1966]
M_LICENSE_USAGE_HISTORY System View [page 1967]
SET SYSTEM LICENSE Statement (System Management) [page 1172]
UNSET SYSTEM LICENSE ALL Statement (System Management) [page 1184]
TEL_LICENSES View (Embedded Statistics Service)
TEL_LICENSE View (Embedded Statistics Service)
View Licenses
License Details
Delete Licenses
Managing SAP HANA Licenses

6.2.151  M_LICENSES System View

Provides information on all of the licenses (if any) that are installed on this system.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARDWARE_KEY</td>
<td>VARCHAR(11)</td>
<td>Displays the hardware key of the SAP HANA installation.</td>
</tr>
<tr>
<td>SYSTEM_ID</td>
<td>VARCHAR(3)</td>
<td>Displays the system identifier (SID).</td>
</tr>
<tr>
<td>INSTALL_NO</td>
<td>VARCHAR(10)</td>
<td>Displays the installation number.</td>
</tr>
<tr>
<td>SYSTEM_NO</td>
<td>VARCHAR(18)</td>
<td>Displays the system number.</td>
</tr>
<tr>
<td>GLAS_APPLICATION_ID</td>
<td>VARCHAR(6)</td>
<td>Displays the license ID for the software product.</td>
</tr>
<tr>
<td>PRODUCT_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the name of the licensed software product, for example, SAP HANA.</td>
</tr>
<tr>
<td>PRODUCT_DESCRIPTION</td>
<td>NVARCHAR(256)</td>
<td>Displays the description of the licensed software product.</td>
</tr>
<tr>
<td>PRODUCT_LIMIT</td>
<td>BIGINT</td>
<td>Displays the allowed product usage as specified by the license in bytes, for example, main memory.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PRODUCT_LIMIT_DESCRIPTION</td>
<td>NVARCHAR(256)</td>
<td>Displays the description of the product usage to be measured and its unit.</td>
</tr>
<tr>
<td>PRODUCT_USAGE</td>
<td>BIGINT</td>
<td>Displays the peak product usage value during last 13 months, measured peri-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>odically, in bytes.</td>
</tr>
<tr>
<td>MEASUREMENT_INTERVAL</td>
<td>INTEGER</td>
<td>Displays the interval of license measurement, in hours.</td>
</tr>
<tr>
<td>FIRST_INSTALLATION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the date of the first installation of the license.</td>
</tr>
<tr>
<td>START_DATE</td>
<td>TIMESTAMP</td>
<td>Displays the start date of the validity period of the license.</td>
</tr>
<tr>
<td>EXPIRATION_DATE</td>
<td>TIMESTAMP</td>
<td>Displays the expiration date of the validity period of the license.</td>
</tr>
<tr>
<td>LAST_SUCCESSFUL_CHECK</td>
<td>TIMESTAMP</td>
<td>Displays the latest date on which the license was successfully checked and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>found valid.</td>
</tr>
<tr>
<td>PERMANENT</td>
<td>VARCHAR(5)</td>
<td>Displays TRUE if the license is permanent and FALSE if the license is tem-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>porary.</td>
</tr>
<tr>
<td>VALID</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the license is valid or not: TRUE/FALSE.</td>
</tr>
<tr>
<td>ENFORCED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the license is a memory-enforced license or not: TRUE/</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE.</td>
</tr>
<tr>
<td>LOCKED_DOWN</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the system is locked down due to license status: TRUE/</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FALSE.</td>
</tr>
<tr>
<td>IS_DATABASE_LOCAL</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the license is a local database local license: TRUE/FALSE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If FALSE, then the license is provided globally by the SystemDB.</td>
</tr>
</tbody>
</table>

**Related Information**

M_LICENSE System View [page 1961]
M_LICENSE_MEASUREMENTS System View [page 1965]
6.2.152 M_LICENSE_MEASUREMENTS System View

Product usage measurements for licensing.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLAS_APPLICATION_ID</td>
<td>VARCHAR(6)</td>
<td>Displays the license ID for the software product.</td>
</tr>
<tr>
<td>MEASURE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the measurement completion time.</td>
</tr>
<tr>
<td>VALUE</td>
<td>BIGINT</td>
<td>Displays the measured value in bytes. This value is NULL when the measure-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ment fails.</td>
</tr>
<tr>
<td>SUCCESSFUL</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the measurement was successful: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

M_LICENSE_MEASUREMENT_STATISTICS System View [page 1966]
M_LICENSE System View [page 1961]
M_LICENSES System View [page 1963]
Managing SAP HANA Licenses
License Keys for SAP HANA Database
6.2.153  M_LICENSE_MEASUREMENT_STATISTICS System View

Collects license measurement statistics from tenant databases.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>YEAR</td>
<td>INTEGER</td>
<td>Displays the year.</td>
</tr>
<tr>
<td>MONTH</td>
<td>INTEGER</td>
<td>Displays the month.</td>
</tr>
<tr>
<td>GLAS_APPLICATION_ID</td>
<td>VARCHAR(6)</td>
<td>Displays the license ID for the software product.</td>
</tr>
<tr>
<td>PEAK_VALUE</td>
<td>BIGINT</td>
<td>Displays the peak value of the application for the given month and year in bytes.</td>
</tr>
</tbody>
</table>

Related Information

M_LICENSE_MEASUREMENTS System View [page 1965]
M_LICENSE_USAGE_HISTORY System View [page 1967]
M_LICENSE System View [page 1961]
M_LICENSES System View [page 1963]
Managing SAP HANA Licenses
License Keys for SAP HANA Database
6.2.154 M_LICENSE_USAGE_HISTORY System View

Information on the maximum resource consumption per time period, used for validity check of license installed on this system (if any).

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT_USAGE</td>
<td>BIGINT</td>
<td>Displays the maximum utilized amount of main memory used in this period in bytes.</td>
</tr>
<tr>
<td>PERIOD_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time of the period that the maximum was used in.</td>
</tr>
<tr>
<td>PERIOD_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the end time of the period the maximum was used in.</td>
</tr>
<tr>
<td>NUM_FAILED_MEASUREMENTS</td>
<td>INTEGER</td>
<td>Displays the number of failed measurements.</td>
</tr>
</tbody>
</table>

**Related Information**

M_LICENSE System View [page 1961]
M_LICENSES System View [page 1963]
M_LICENSE_MEASUREMENTS System View [page 1965]
M_LICENSE_MEASUREMENT_STATISTICS System View [page 1966]
License Details
## 6.2.155 M_LIVECACHE_CONTAINER_STATISTICS System View

Provides accumulated liveCache container statistics.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID.</td>
</tr>
<tr>
<td>OMS_CLASS_ID</td>
<td>INTEGER</td>
<td>Displays the OMS class ID.</td>
</tr>
<tr>
<td>OMS_CLASS_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the OMS class name.</td>
</tr>
<tr>
<td>OMS_SCHEMA_HANDLE</td>
<td>INTEGER</td>
<td>Displays the OMS schema handle.</td>
</tr>
<tr>
<td>OMS_CONTAINER_NO</td>
<td>INTEGER</td>
<td>Displays the OMS container number.</td>
</tr>
<tr>
<td>CLASS_SIZE</td>
<td>BIGINT</td>
<td>Displays the object class size in bytes.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(3)</td>
<td>Displays the object type: STD, KEY, or VAR.</td>
</tr>
<tr>
<td>KEY_SIZE</td>
<td>INTEGER</td>
<td>Displays the object key size in bytes.</td>
</tr>
<tr>
<td>KEY_PARTITION_COUNT</td>
<td>INTEGER</td>
<td>Displays the key partition count.</td>
</tr>
<tr>
<td>KEY_PARTITION_PREFIX_SIZE</td>
<td>INTEGER</td>
<td>Displays the key partition prefix length in bytes.</td>
</tr>
<tr>
<td>CACHED_KEYS</td>
<td>VARCHAR(5)</td>
<td>Displays whether the cached keys are active: TRUE/FALSE.</td>
</tr>
<tr>
<td>COPY_ON_UPDATE</td>
<td>VARCHAR(5)</td>
<td>Displays whether copy on update is active: TRUE/FALSE.</td>
</tr>
<tr>
<td>PREFETCH</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the prefetch is active: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_LOADED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the statistics are loaded into the memory: TRUE/FALSE.</td>
</tr>
<tr>
<td>OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the object count.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OBJECT_SIZE_SUM</td>
<td>BIGINT</td>
<td>Displays the object size sum in bytes.</td>
</tr>
<tr>
<td>GC_PENDING_OBJECT_DELETE_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of objects that are deleted and not yet freed by the garbage collection.</td>
</tr>
<tr>
<td>GC_PENDING_OBJECT_DELETE_SIZE_SUM</td>
<td>BIGINT</td>
<td>Displays the transient size sum of the objects that are deleted and not yet freed by the garbage collection in bytes.</td>
</tr>
<tr>
<td>HISTORY_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of transient history object versions that are not yet freed by the garbage collection.</td>
</tr>
<tr>
<td>HISTORY_SIZE_SUM</td>
<td>BIGINT</td>
<td>Displays the size sum, in bytes, of the transient history object versions that are not yet freed by the garbage collection.</td>
</tr>
<tr>
<td>PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the page count.</td>
</tr>
<tr>
<td>PAGE_SIZE_SUM</td>
<td>BIGINT</td>
<td>Displays the page size sum in bytes.</td>
</tr>
<tr>
<td>OBJECT_CREATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of objects created since the last restart.</td>
</tr>
<tr>
<td>OBJECT_UPDATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of objects updated since the last restart.</td>
</tr>
<tr>
<td>OBJECT_DELETE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of objects delete since the last restart.</td>
</tr>
<tr>
<td>HEAP_USAGE</td>
<td>BIGINT</td>
<td>Displays the memory currently used by the container in bytes.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the created timestamp.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view has a resettable counterpart; you can see the values since the last reset in the M_LIVECACHE_CONTAINER_STATISTICS_RESET system view. To reset the view, execute the following statement:

```sql
ALTER SYSTEM RESET MONITORING VIEW SYS.M_LIVECACHE_CONTAINER_STATISTICS_RESET;
```
6.2.156 M_LIVECACHE_CONTAINER_STATISTICS_RESET System View

Provides accumulated LiveCache container statistics since last reset.

This view contains values accumulated since the last reset of the main view M_LIVECACHE_CONTAINER_STATISTICS. Refer to M_LIVECACHE_CONTAINER_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_LIVECACHE_CONTAINER_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_LIVECACHE_CONTAINER_STATISTICS System View [page 1968]

6.2.157 M_LIVECACHE_LOCKS System View

Detailed information on the Object Management System (OMS) locks.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
ID | VARCHAR(20) | Displays the ID of the lock: OID, container ID, or schema ID, depending on the lock class.
CONTAINER_ID | BIGINT | Displays the ID of the corresponding container.
CLASS | VARCHAR(20) | Displays the class that the lock belongs to.
GRANTED_MODE | VARCHAR(20) | Displays the mode that the lock has granted to some request(s).
MODE | VARCHAR(20) | Displays the mode that the request is requesting on the lock: Free, IntendShare, IntendExclusive, Share, ShareIntendExclusive, or Exclusive.
TYPE | VARCHAR(10) | Displays when the lock can be removed, either at transaction-end (EOT) or if not visible anymore (Consistent).
STATE | VARCHAR(10) | Displays the state of the lock request.
TID | BIGINT | Displays the transaction ID belonging to this lock request.
TIMEOUT | BIGINT | Displays remaining timeout, if a timeout is specified.

### Additional Information

The liveCache uses its own lock-manager for object locks, container locks, and schema locks. This view shows information about all the locks, which are currently kept in this lock manager.

Exclusive locks on objects are in most cases not managed by this lock manager, but the necessary lock information is stored in the respective object header.

This view can only be used if liveCache is enabled.

### Related Information

- M_LIVECACHE_LOCK_STATISTICS_RESET System View [page 1973]
- M_LIVECACHE_LOCK_STATISTICS System View [page 1972]
6.2.158 M_LIVECACHE_LOCK_STATISTICS System View

Provides accumulated LiveCache lock statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CONTAINER_EXCLUSIVE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of exclusive lock requests on containers.</td>
</tr>
<tr>
<td>CONTAINER.Shared_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of shared lock requests on containers.</td>
</tr>
<tr>
<td>CONTAINER_COLLISION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of lock collisions on containers.</td>
</tr>
<tr>
<td>CONTAINER_TIMEOUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of lock timeouts on containers.</td>
</tr>
<tr>
<td>SCHEMA_EXCLUSIVE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of exclusive lock requests on schemas.</td>
</tr>
<tr>
<td>SCHEMA_SHARED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of shared lock requests on schemas.</td>
</tr>
<tr>
<td>SCHEMA_COLLISION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of lock collisions on schemas.</td>
</tr>
<tr>
<td>SCHEMA_TIMEOUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of lock timeouts on schemas.</td>
</tr>
<tr>
<td>OBJECT_EXCLUSIVE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of exclusive lock requests on objects.</td>
</tr>
<tr>
<td>OBJECT_SHARED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of shared lock requests on objects.</td>
</tr>
<tr>
<td>OBJECT_COLLISION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of lock collisions on objects.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td>OBJECT_TIMEOUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of timeouts on objects.</td>
</tr>
<tr>
<td>COMMITTED_REQUESTS_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of committed requests.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view is only usable if liveCache is enabled.

The liveCache uses its own lock-manager for object locks, container locks, and schema locks. This view shows the accumulated statistics of the lock requests to this lock-manager, which have been executed since restart.

This view has a resettable counterpart; you can see the values since the last reset in the M_LIVECACHE_LOCK_STATISTICS_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_LIVECACHE_LOCK_STATISTICS_RESET;
```

**Related Information**

- M_LIVECACHE_LOCK_STATISTICS_RESET System View [page 1973]
- M_LIVECACHE_LOCKS System View [page 1970]
- LOCK TABLE Statement (Transaction Management) [page 1056]

6.2.159 M_LIVECACHE_LOCK_STATISTICS_RESET System View

LiveCache lock statistics (since last reset).

This view contains values accumulated since the last reset of the main view M_LIVECACHE_LOCK_STATISTICS. Refer to M_LIVECACHE_LOCK_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_LIVECACHE_LOCK_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

**Related Information**

- M_LIVECACHE_LOCK_STATISTICS System View [page 1972]
6.2.160 M_LIVECACHE_OMS_VERSIONS System View

Detailed information on the OMS versions that currently exists.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VERSION_ID</td>
<td>VARCHAR(22)</td>
<td>Displays the ID of the OMS version.</td>
</tr>
<tr>
<td>CREATE_DATE</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of the creation of the OMS version.</td>
</tr>
<tr>
<td>LAST_OPEN_DATE</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the OMS version was opened last.</td>
</tr>
<tr>
<td>IS_OPEN</td>
<td>VARCHAR(5)</td>
<td>Displays the flag indicating whether the version is currently open: TRUE/FALSE.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the connection/session that the version is bound to.</td>
</tr>
<tr>
<td>LIVECACHE_MVCC_SNAPSHOT_TIME-</td>
<td>BIGINT</td>
<td>Displays the consistent view MVCC snapshot timestamp.</td>
</tr>
<tr>
<td>STAMP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEAP_USAGE</td>
<td>BIGINT</td>
<td>Displays the memory that is currently used by the version in bytes.</td>
</tr>
<tr>
<td>VERSION_DESCRIPTION</td>
<td>NVARCHAR(512)</td>
<td>Displays the description.</td>
</tr>
</tbody>
</table>

**Additional Information**

Status information is shown for each OMS version that currently exists.
This view can only be used if liveCache is enabled.

**Related Information**

HOST_LIVECACHE_OMS_VERSIONS View (Embedded Statistics Service)
M_LIVECACHE_PROCEDURE_STATISTICS System View [page 1975]
System-Versioned Tables

6.2.161 M_LIVECACHE_PROCEDURE_STATISTICS System View

Provides accumulated LiveCache procedure statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>METHOD_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the method name.</td>
</tr>
<tr>
<td>CALL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of calls of the pro-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cedure.</td>
</tr>
<tr>
<td>SUM_RUN_TIME</td>
<td>BIGINT</td>
<td>Displays the total runtime of the proce-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dure in microseconds.</td>
</tr>
<tr>
<td>MAX_RUN_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum runtime of the proce-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dure in microseconds.</td>
</tr>
<tr>
<td>MIN_RUN_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum runtime of the proce-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dure in microseconds.</td>
</tr>
<tr>
<td>AVERAGE_RUN_TIME</td>
<td>BIGINT</td>
<td>Displays the average runtime of the proce-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dure in microseconds.</td>
</tr>
<tr>
<td>DEREF_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of OID derefs.</td>
</tr>
<tr>
<td>DEREF_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of OID derefs against basis.</td>
</tr>
<tr>
<td>DEREF_BASE_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of OID derefs against basis from within an OMS version.</td>
</tr>
<tr>
<td>DEREF_KEYED_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of key derefs.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DEREF_KEYED_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of key derefs against basis.</td>
</tr>
<tr>
<td>DEREF_KEYED_OBJECT_BASE_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of key derefs against basis from within an OMS version.</td>
</tr>
<tr>
<td>ITER_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of objects read via an OID-iterator from the basis.</td>
</tr>
<tr>
<td>ITER_KEYED_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of objects read via a key-iterator from the basis.</td>
</tr>
<tr>
<td>ITER_BASE_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of objects read via an OID-iterator from the basis from within an OMS version.</td>
</tr>
<tr>
<td>ITER_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of objects, created in an OMS-version, read via an OID-iterator.</td>
</tr>
<tr>
<td>ITER_KEYED_OBJECT_BASE_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of objects read via an OID-iterator from the basis from within an OMS version.</td>
</tr>
<tr>
<td>ITER_KEYED_OBJECT_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of objects, created in an OMS version, read via a key-iterator.</td>
</tr>
<tr>
<td>DEREF_VAR_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of derefs of var objects.</td>
</tr>
<tr>
<td>DEREF_VAR_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of derefs of var objects against basis.</td>
</tr>
<tr>
<td>DEREF_VAR_OBJECT_BASE_SIZE</td>
<td>BIGINT</td>
<td>Displays the accumulated size of var objects read from the basis in bytes.</td>
</tr>
<tr>
<td>DEREF_VAR_OBJECT_BASE_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of derefs of var objects against basis from within an OMS version.</td>
</tr>
<tr>
<td>DEREF_VAR_OBJECT_BASE_IN_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the accumulated size of var objects read from the basis from within an OMS version in bytes.</td>
</tr>
<tr>
<td>NEW_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of newly created standard objects.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NEW_KEYED_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of newly created keyed objects.</td>
</tr>
<tr>
<td>NEW_VAR_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of newly created var objects.</td>
</tr>
<tr>
<td>NEW_OBJECT_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of newly created standard objects in an OMS version.</td>
</tr>
<tr>
<td>NEW_KEYED_OBJECT_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of newly created keyed objects in an OMS version.</td>
</tr>
<tr>
<td>NEW_VAR_OBJECT_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of newly created var objects in an OMS version.</td>
</tr>
<tr>
<td>STORE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of store calls on standard objects.</td>
</tr>
<tr>
<td>STORE_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of store calls on standard objects in an OMS version.</td>
</tr>
<tr>
<td>STORE_KEYED_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of store calls on keyed objects.</td>
</tr>
<tr>
<td>STORE_KEYED_OBJECT_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of store calls on keyed objects in an OMS version.</td>
</tr>
<tr>
<td>STORE_VAR_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of store calls on var objects.</td>
</tr>
<tr>
<td>STORE_VAR_OBJECT_SIZE</td>
<td>BIGINT</td>
<td>Displays the accumulated size of store calls on var objects in bytes.</td>
</tr>
<tr>
<td>STORE_VAR_OBJECT_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of store calls on var objects in an OMS version.</td>
</tr>
<tr>
<td>STORE_VAR_OBJECT_IN_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the accumulated size of store calls on var objects in an OMS version in bytes.</td>
</tr>
<tr>
<td>STORE_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of updates of standard objects in the basis.</td>
</tr>
<tr>
<td>STORE_KEYED_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of updates of keyed objects in the basis.</td>
</tr>
<tr>
<td>STORE_VAR_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of updates of var objects in the basis.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>STORE_VAR_OBJECT_BASE_SIZE</td>
<td>BIGINT</td>
<td>Displays the accumulated sizes of updated objects in the basis in bytes.</td>
</tr>
<tr>
<td>DELETE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of delete calls on standard objects.</td>
</tr>
<tr>
<td>DELETE_KEYED_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of delete calls on keyed objects.</td>
</tr>
<tr>
<td>DELETE_VAR_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of delete calls on var objects.</td>
</tr>
<tr>
<td>DELETE_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of delete calls on standard objects in an OMS version.</td>
</tr>
<tr>
<td>DELETE_KEYED_OBJECT_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of delete calls on keyed objects in an OMS version.</td>
</tr>
<tr>
<td>DELETE_VAR_OBJECT_IN_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of delete calls on var objects in an OMS version.</td>
</tr>
<tr>
<td>DELETE_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deleted standard objects in the basis.</td>
</tr>
<tr>
<td>DELETE_KEYED_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deleted keyed objects in the basis.</td>
</tr>
<tr>
<td>DELETE_VAR_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deleted var objects in the basis.</td>
</tr>
<tr>
<td>LOCK_EXCLUSIVE_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of exclusive lock requests on standard objects in the basis.</td>
</tr>
<tr>
<td>LOCK_SHARE_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of share lock requests on standard objects in the basis.</td>
</tr>
<tr>
<td>LOCK_EXCLUSIVE_KEYED_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of exclusive lock requests on keyed objects in the basis.</td>
</tr>
<tr>
<td>LOCK_SHARE_KEYED_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of share lock requests on keyed objects in the basis.</td>
</tr>
<tr>
<td>LOCK_EXCLUSIVE_VAR_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of exclusive lock requests on var objects in the basis.</td>
</tr>
<tr>
<td>LOCK_SHARE_VAR_OBJECT_BASE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of share lock requests on var objects in the basis.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RELEASE_CALLED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of OMS release calls.</td>
</tr>
<tr>
<td>RELEASE_EXECUTED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of released standard objects.</td>
</tr>
<tr>
<td>RELEASE_EXECUTED_KEYED_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of releases keyed objects.</td>
</tr>
<tr>
<td>RELEASE_EXECUTED_VAR_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of releases var objects.</td>
</tr>
<tr>
<td>HISTORY_HOP_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of hops in the history chain during deref.</td>
</tr>
<tr>
<td>ITER_HISTORY_HOP_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of hops in the history chain during iteration.</td>
</tr>
<tr>
<td>exception_count</td>
<td>BIGINT</td>
<td>Displays the number of dbp-exceptions thrown.</td>
</tr>
<tr>
<td>OUT_OF_DATE_EXCEPTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of out-of-date-exceptions thrown.</td>
</tr>
<tr>
<td>OUT_OF_MEMORY_EXCEPTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of out-of-memory-exceptions thrown.</td>
</tr>
<tr>
<td>TIMEOUT_EXCEPTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of timeout-exceptions thrown.</td>
</tr>
<tr>
<td>OMS_TERMINATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of OMS terminate calls.</td>
</tr>
<tr>
<td>SUBTRANSACTION_ROLLBACK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rolled back subtransactions.</td>
</tr>
<tr>
<td>SUBTRANSACTION_COMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of committed subtransactions.</td>
</tr>
<tr>
<td>MAX_SUBTRANSACTION_LEVEL</td>
<td>BIGINT</td>
<td>Displays the maximum subtransaction level.</td>
</tr>
<tr>
<td>NEW_CONSISTENT_VIEW_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of new-consistent-view calls with objects.</td>
</tr>
<tr>
<td>SUM_NEW_CONSISTENT_VIEW_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of the wait times of the new-consistent-view calls with objects.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AVERAGE_NEW_CONSISTENT_VIEW_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the average wait time of new-consistent-view calls with objects in seconds.</td>
</tr>
<tr>
<td>NEW_CONSISTENT_VIEW_MAX_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the maximal wait time of new-consistent-view calls with objects in seconds.</td>
</tr>
<tr>
<td>KEY_CACHE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of cache hits in the key-cache.</td>
</tr>
<tr>
<td>KEY_MISS_CACHE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of cache hits in the key-miss-cache.</td>
</tr>
<tr>
<td>DEREF_VERSION_KEYED_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of key-derefs in an OMS version on objects created in this version.</td>
</tr>
<tr>
<td>OMS_REHASH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rehashes of the OMS-object-hash.</td>
</tr>
<tr>
<td>AVERAGE_HASHCHAIN_SEARCH_LENGTH</td>
<td>BIGINT</td>
<td>Displays the average search length on the hash-chains, in microseconds, of the OMS-object-hash.</td>
</tr>
<tr>
<td>MAX_HASHCHAIN_LENGTH</td>
<td>BIGINT</td>
<td>Displays the maximum length of a hash-chain of the OMS-object-hash.</td>
</tr>
<tr>
<td>VERSION_CREATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of created OMS versions.</td>
</tr>
<tr>
<td>VERSION_OPEN_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of open-version calls.</td>
</tr>
<tr>
<td>VERSION_CLOSE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of close-version calls.</td>
</tr>
<tr>
<td>VERSION_DROP_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of dropped OMS versions.</td>
</tr>
<tr>
<td>USER_ALLOC_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of allocations via a user allocator.</td>
</tr>
<tr>
<td>USER_MAX_CHUNK_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of a chunk allocated via a user allocator in bytes.</td>
</tr>
<tr>
<td>USER_MIN_CHUNK_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of a chunk allocated via a user allocator in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>USER_AVERAGE_CHUNK_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of a chunk allocated via a user allocator in bytes.</td>
</tr>
<tr>
<td>USER_DELETE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deallocations via a user allocator.</td>
</tr>
<tr>
<td>USER_MAX_CHUNK_DELETED_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of a chunk deallocated via a user allocator in bytes.</td>
</tr>
<tr>
<td>USER_MIN_CHUNK_DELETED_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of a chunk deallocated via a user allocator in bytes.</td>
</tr>
<tr>
<td>USER_AVERAGE_CHUNK_DELETED_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of a chunk deallocated via a user allocator in bytes.</td>
</tr>
<tr>
<td>USER_DELTA_MAX_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximal difference in size between allocation and deallocation on a user allocator during the execution of a method in bytes.</td>
</tr>
<tr>
<td>OMS_ALLOC_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of allocations via an OMS internal allocator.</td>
</tr>
<tr>
<td>OMS_MAX_CHUNK_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of a chunk allocated via an OMS internal allocator in bytes.</td>
</tr>
<tr>
<td>OMS_MIN_CHUNK_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of a chunk allocated via an OMS internal allocator in bytes.</td>
</tr>
<tr>
<td>OMS_AVERAGE_CHUNK_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of a chunk allocated via an OMS internal allocator in bytes.</td>
</tr>
<tr>
<td>OMS_DELETE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deallocations via an OMS internal allocator.</td>
</tr>
<tr>
<td>OMS_MAX_CHUNK_DELETED_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of a chunk deallocated via an OMS internal allocator in bytes.</td>
</tr>
<tr>
<td>OMS_MIN_CHUNK_DELETED_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of a chunk deallocated via an OMS internal allocator in bytes.</td>
</tr>
<tr>
<td>OMS_AVERAGE_CHUNK_DELETED_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of a chunk deallocated via an OMS internal allocator in bytes.</td>
</tr>
</tbody>
</table>

SAP HANA SQL Reference Guide for SAP HANA Platform
System Views Reference
PUBLIC 1981
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMS_DELTA_MAX_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum difference in size between allocation and deallocation on an OMS internal allocator during the execution of a method in bytes.</td>
</tr>
<tr>
<td>LAST_USER_DELTA_MAX_SIZE</td>
<td>BIGINT</td>
<td>Displays the last peak delta of the user heap consumption during procedure execution in bytes.</td>
</tr>
<tr>
<td>MAX_USER_DELTA_MAX_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum peak delta of the user heap consumption during procedure execution in bytes.</td>
</tr>
<tr>
<td>MIN_USER_DELTA_MAX_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum peak delta of the user heap consumption during procedure execution in bytes.</td>
</tr>
<tr>
<td>SUM_USER_DELTA_MAX_SIZE</td>
<td>BIGINT</td>
<td>Displays the total peak delta of the user heap consumption during procedure execution in bytes.</td>
</tr>
<tr>
<td>AVG_USER_DELTA_MAX_SIZE</td>
<td>BIGINT</td>
<td>Displays the peak delta of the user heap consumption during procedure execution in bytes.</td>
</tr>
<tr>
<td>LAST_OMS_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the last delta of the OMS heap consumption between start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>MAX_OMS_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum delta of the OMS heap consumption between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>MIN_OMS_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum delta of the OMS heap consumption between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>SUM_OMS_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the total delta of the OMS heap consumption between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>AVG_OMS_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the average delta of the OMS heap consumption between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LAST_VERSION_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the last delta of the OMS version heap consumption between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>MAX_VERSION_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum delta of the OMS version heap consumption between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>MIN_VERSION_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum delta of the OMS version heap consumption between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>SUM_VERSION_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the total delta of the OMS version heap consumption between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>AVG_VERSION_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the average delta of the OMS version heap consumption between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>LAST_HEAP_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the last delta of the overall heap consumption (user + OMS + OMS version) between the start and end of the procedure execution.</td>
</tr>
<tr>
<td>MAX_HEAP_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum delta of the overall heap consumption (user + OMS + OMS version) between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>MIN_HEAP_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum delta of the overall heap consumption (user + OMS + OMS version) between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>SUM_HEAP_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the total delta of the overall heap consumption (user + OMS + OMS version) between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AVG_HEAP_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the average delta of the overall heap consumption (user + OMS + OMS version) between the start and end of the procedure execution in bytes.</td>
</tr>
<tr>
<td>STREAM_COMMUNICATION_TIME</td>
<td>BIGINT</td>
<td>Displays the total stream communication time in microseconds.</td>
</tr>
<tr>
<td>MAX_STREAM_COMMUNICATION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum stream communication time in microseconds.</td>
</tr>
<tr>
<td>MIN_STREAM_COMMUNICATION_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum stream communication time in microseconds.</td>
</tr>
<tr>
<td>AVG_STREAM_COMMUNICATION_TIME</td>
<td>BIGINT</td>
<td>Displays the average stream communication time in microseconds.</td>
</tr>
<tr>
<td>STREAM_READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of communications for reading ABAP tables.</td>
</tr>
<tr>
<td>STREAM_WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of communications for writing ABAP tables.</td>
</tr>
<tr>
<td>STREAM_READ_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows read from ABAP tables.</td>
</tr>
<tr>
<td>STREAM_WRITE_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows written to ABAP tables.</td>
</tr>
<tr>
<td>LAST_INTERNAL_SQL_PREPARE_TIME</td>
<td>BIGINT</td>
<td>Displays last the internal SQL prepare time.</td>
</tr>
<tr>
<td>MAX_INTERNAL_SQL_PREPARE_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum internal SQL prepare time.</td>
</tr>
<tr>
<td>MIN_INTERNAL_SQL_PREPARE_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum internal SQL prepare time.</td>
</tr>
<tr>
<td>SUM_INTERNAL_SQL_PREPARE_TIME</td>
<td>BIGINT</td>
<td>Displays the total internal SQL prepare time.</td>
</tr>
<tr>
<td>AVG_INTERNAL_SQL_PREPARE_TIME</td>
<td>BIGINT</td>
<td>Displays the average internal SQL prepare time.</td>
</tr>
<tr>
<td>LAST_INTERNAL_SQL_EXECUTE_TIME</td>
<td>BIGINT</td>
<td>Displays the last internal SQL execute time.</td>
</tr>
<tr>
<td>MAX_INTERNAL_SQL_EXECUTE_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum internal SQL execute time.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>MIN_INTERNAL_SQL_EXECUTE_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum internal SQL execute time.</td>
</tr>
<tr>
<td>SUM_INTERNAL_SQL_EXECUTE_TIME</td>
<td>BIGINT</td>
<td>Displays the total internal SQL execute time.</td>
</tr>
<tr>
<td>AVG_INTERNAL_SQL_EXECUTE_TIME</td>
<td>BIGINT</td>
<td>Displays the average internal SQL execute time.</td>
</tr>
<tr>
<td>LAST_INTERNAL_SQL_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the last internal SQL fetch time.</td>
</tr>
<tr>
<td>MAX_INTERNAL_SQL_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum internal SQL fetch time.</td>
</tr>
<tr>
<td>MIN_INTERNAL_SQL_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum internal SQL fetch time.</td>
</tr>
<tr>
<td>SUM_INTERNAL_SQL_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the total internal SQL fetch time.</td>
</tr>
<tr>
<td>AVG_INTERNAL_SQL_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the avg internal SQL fetch time.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view can only be used if liveCache is enabled.

For each liveCache procedure, which has been called already since the last restart, statistics are shown.

This view has a resettable counterpart; you can see the values since the last reset in the M_LIVECACHE_PROCEDURE_STATISTICS_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_LIVECACHE_PROCEDURE_STATISTICS_RESET;
```

**Related Information**

- M_LIVECACHE_PROCEDURE_STATISTICS_RESET System View [page 1986]
- ALTER SYSTEM RESET MONITORING VIEW Statement (System Management) [page 565]
- STATISTICS_OBJECTS Table (Embedded Statistics Service)
- The Statistics Service
- Managing Statistics
- Procedures
6.2.162 M_LIVECACHE_PROCEDURE_STATISTICS_RESET
System View

LiveCache procedure statistics (since last reset).

This view contains values accumulated since the last reset of the main view M_LIVECACHE_PROCEDURE_STATISTICS. Refer to M_LIVECACHE_PROCEDURE_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_LIVECACHE_PROCEDURE_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_LIVECACHE_PROCEDURE_STATISTICS System View [page 1975]

6.2.163 M_LIVECACHE_SCHEMA_STATISTICS System View

Provides accumulated liveCache schema statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>OMS_SCHEMA_HANDLE</td>
<td>INTEGER</td>
<td>Displays the OMS schema ID.</td>
</tr>
<tr>
<td>OMS_SCHEMA_NAME</td>
<td>NVARCHAR(96)</td>
<td>Displays the OMS schema name.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the created timestamp.</td>
</tr>
</tbody>
</table>

Additional Information

This view has a resettable counterpart; you can see the values since the last reset in the M_LIVECACHE_SCHEMA_STATISTICS_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_LIVECACHE_SCHEMA_STATISTICS_RESET;
```
6.2.164 M_LIVECACHE_SCHEMA_STATISTICS_RESET System View

Provides accumulated LiveCache schema statistics since the last reset.

This view contains values accumulated since the last reset of the main view
M_LIVECACHE_SCHEMA_STATISTICS. Refer to M_LIVECACHE_SCHEMA_STATISTICS for information about
the structure and use of this view.

In addition to the members mentioned in M_LIVECACHE_SCHEMA_STATISTICS, this view also contains a
timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_LIVECACHE_SCHEMA_STATISTICS System View [page 1986]

6.2.165 M_LOAD_HISTORY_HOST System View

Host specific load history KPIs.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>TIME</td>
<td>TIMESTAMP</td>
<td>Displays the KPI collection timestamp.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CPU</td>
<td>BIGINT</td>
<td>Displays the percent of CPU used by all processes.</td>
</tr>
<tr>
<td>MEMORY_RESIDENT</td>
<td>BIGINT</td>
<td>Displays the physical memory used for all SAP HANA processes in bytes.</td>
</tr>
<tr>
<td>MEMORY_TOTAL_RESIDENT</td>
<td>BIGINT</td>
<td>Displays the physical memory used for all processes in bytes.</td>
</tr>
<tr>
<td>MEMORY_USED</td>
<td>BIGINT</td>
<td>Displays the memory used for all SAP HANA processes in bytes.</td>
</tr>
<tr>
<td>MEMORY_ALLOCATION_LIMIT</td>
<td>BIGINT</td>
<td>Displays the memory allocation limit for all processes of an SAP HANA instance in bytes.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the physical memory size in bytes.</td>
</tr>
<tr>
<td>DISK_USED</td>
<td>BIGINT</td>
<td>Displays the amount of disk used in bytes.</td>
</tr>
<tr>
<td>DISK_SIZE</td>
<td>BIGINT</td>
<td>Displays the disk size in bytes.</td>
</tr>
<tr>
<td>NETWORK_IN</td>
<td>BIGINT</td>
<td>Displays the bytes read from network by all processes in bytes per sample.</td>
</tr>
<tr>
<td>NETWORK_OUT</td>
<td>BIGINT</td>
<td>Displays the bytes written to network by all processes in bytes per sample.</td>
</tr>
<tr>
<td>SWAP_IN</td>
<td>BIGINT</td>
<td>Displays the bytes read from swap by all processes in bytes per sample.</td>
</tr>
<tr>
<td>SWAP_OUT</td>
<td>BIGINT</td>
<td>Displays the bytes written to swap by all processes in bytes per sample.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_LOAD_HISTORY_INFO System View [page 1989]
- M_LOAD_HISTORY_SERVICE System View [page 1990]
- LOAD Statement (Data Manipulation) [page 1054]
- View Change History
- PERSISTENCE_HISTORY System View [page 1608]
6.2.166  M_LOAD_HISTORY_INFO System View

Load history KPI description.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view name where the KPI can be found.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name where the KPI can be found.</td>
</tr>
<tr>
<td>IS_CUMULATIVE</td>
<td>VARCHAR(5)</td>
<td>Displays the cumulative KPIs that return a relative value since the previous sample. Noncumulative KPIs return an absolute value for the current point in time.</td>
</tr>
<tr>
<td>SQL_SOURCE</td>
<td>VARCHAR(256)</td>
<td>Displays the equivalent SQL statement.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(256)</td>
<td>Displays the KPI description.</td>
</tr>
<tr>
<td>SAMPLE_UNIT</td>
<td>VARCHAR(16)</td>
<td>Displays the sample unit.</td>
</tr>
<tr>
<td>DISPLAY_HIERARCHY</td>
<td>VARCHAR(8)</td>
<td>Displays the recommended display hierarchy/sorting criteria.</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the recommended display name.</td>
</tr>
<tr>
<td>DISPLAY_UNIT</td>
<td>VARCHAR(16)</td>
<td>Displays the recommended display unit.</td>
</tr>
<tr>
<td>DISPLAY_LINE_COLOR</td>
<td>INTEGER</td>
<td>Displays the recommended display line color as RGB.</td>
</tr>
<tr>
<td>DISPLAY_LINE_STYLE</td>
<td>TINYINT</td>
<td>Displays the recommended display line style: 1=solid, 2=dotted, or 3=dashed.</td>
</tr>
<tr>
<td>DISPLAY_DIVIDER</td>
<td>INTEGER</td>
<td>Displays the divider to convert SAMPLE_UNIT in DISPLAY_UNIT.</td>
</tr>
<tr>
<td>DISPLAY_Y_SCALE</td>
<td>TINYINT</td>
<td>Indicates that the KPIs with the same value should be shown with the same Y scale in a chart. Special value 1 is used for CPU KPIs with a fixed Y scale of 100.</td>
</tr>
</tbody>
</table>
### 6.2.167 M_LOAD_HISTORY_SERVICE System View

Lists service-specific load history KPIs.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>TIME</td>
<td>TIMESTAMP</td>
<td>Displays the KPI collection timestamp.</td>
</tr>
<tr>
<td>CPU</td>
<td>BIGINT</td>
<td>Displays the percent of CPU used by the service.</td>
</tr>
<tr>
<td>SYSTEM_CPU</td>
<td>BIGINT</td>
<td>Displays the percent of OS Kernel/System CPU used by the service.</td>
</tr>
<tr>
<td>MEMORY_USED</td>
<td>BIGINT</td>
<td>Displays the memory used by the service in bytes.</td>
</tr>
<tr>
<td>MEMORY_ALLOCATION_LIMIT</td>
<td>BIGINT</td>
<td>Displays the memory allocation limit for the service in bytes.</td>
</tr>
<tr>
<td>HANDLE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of open handles.</td>
</tr>
<tr>
<td>PING_TIME</td>
<td>BIGINT</td>
<td>Displays the duration, in microseconds, of the service ping request (THREAD_METHOD=__nsWatchdog). This request includes the time it takes to measure the values shown in this view.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SWAP_IN</td>
<td>BIGINT</td>
<td>Displays the bytes read from the swap by the service (column <code>12(majflt)</code> in <code>/proc/&lt;pid&gt;/stat * sysconf(_SC_PAGE_SIZE)</code>,)</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CS_WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of write requests (INSERT, UPDATE, DELETE).</td>
</tr>
<tr>
<td>CS_MERGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of merge requests.</td>
</tr>
<tr>
<td>CS_UNLOAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of column unloads.</td>
</tr>
<tr>
<td>ACTIVE_THREAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active threads (THREAD_STATE != 'inactive').</td>
</tr>
<tr>
<td>WAITING_THREAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of waiting threads.</td>
</tr>
<tr>
<td>TOTAL_THREAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of threads.</td>
</tr>
<tr>
<td>ACTIVE_SQL_EXECUTOR_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active SQLExecutors.</td>
</tr>
<tr>
<td>WAITING_SQL_EXECUTOR_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of waiting SQLExecutors.</td>
</tr>
<tr>
<td>TOTAL_SQL_EXECUTOR_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of SQLExecutors.</td>
</tr>
<tr>
<td>DATA_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the bytes written to the data area.</td>
</tr>
<tr>
<td>DATA_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the time used for writing to the data area in microseconds.</td>
</tr>
<tr>
<td>LOG_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the bytes written to the log area.</td>
</tr>
<tr>
<td>LOG_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the time used for writing to the log area in microseconds.</td>
</tr>
<tr>
<td>DATA_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the bytes read from the data area.</td>
</tr>
<tr>
<td>DATA_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the time used for reading from the data area in microseconds.</td>
</tr>
<tr>
<td>LOG_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the bytes read from the log area.</td>
</tr>
<tr>
<td>LOG_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the time used for reading from the log area in microseconds.</td>
</tr>
<tr>
<td>DATA_BACKUP_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the bytes written to the data backup.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DATA_BACKUP_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the time used for writing to the data backup in microseconds.</td>
</tr>
<tr>
<td>LOG_BACKUP_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the bytes written to the log backup.</td>
</tr>
<tr>
<td>LOG_BACKUP_WRITE_TIME</td>
<td>BIGINT</td>
<td>The time used for writing to the log backup.</td>
</tr>
<tr>
<td>MUTEX_COLLISION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of collisions on mutexes.</td>
</tr>
<tr>
<td>READ_WRITE_LOCK_COLLISION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of collisions on read/write locks.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_ADMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of admission requests admitted by admission control.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_REJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of session requests rejected by admission control.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_QUEUE_SIZE</td>
<td>BIGINT</td>
<td>Displays the number of session requests waiting in the admission control queue.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total wait time of the session requests queued in the admission control queue in microseconds.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_ENQUEUE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of session requests queued by admission control.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_DEQUEUE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of session requests dequeued for deferred execution by admission control.</td>
</tr>
<tr>
<td>ADMISSION_CONTROL_TIME_OUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of session requests dequeued for timed out by admission control.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_LOAD_HISTORY_HOST System View [page 1987]
- M_LOAD_HISTORY_INFO System View [page 1989]
- LOAD Statement (Data Manipulation) [page 1054]
- View Change History
6.2.168  M_LOCK_WAITS_STATISTICS System View

Provides the accumulated lock wait count and duration for record locks, table locks, and metadata locks for all available services from database start up until the current time.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>LOCK_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the lock type: RECORD/ TABLE.</td>
</tr>
<tr>
<td>TOTAL_LOCK_WAITS</td>
<td>BIGINT</td>
<td>Displays the total lock wait count.</td>
</tr>
<tr>
<td>TOTAL_LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total lock wait duration in microseconds.</td>
</tr>
</tbody>
</table>

Related Information

LOCK TABLE Statement (Transaction Management) [page 1056]
Thread Details
6.2.169  M_LOG_BUFFERS System View

Provides information about log buffer statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>LOG_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the log mode.</td>
</tr>
<tr>
<td>BUFFER_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of one log buffer in memory in kilobytes.</td>
</tr>
<tr>
<td>BUFFER_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of log buffers in memory.</td>
</tr>
<tr>
<td>SEGMENT_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of one log segment in megabytes.</td>
</tr>
<tr>
<td>BACKUP_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the log segment backup is enabled: TRUE, FALSE (FALSE on log backup history broken).</td>
</tr>
<tr>
<td>BACKUP_TIMEOUT</td>
<td>BIGINT</td>
<td>Displays the log segment backup timeout in seconds.</td>
</tr>
<tr>
<td>SWITCH_NOWAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of buffer switches without blocking on buffer semaphore.</td>
</tr>
<tr>
<td>SWITCH_WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of buffer switches with blocking on buffer semaphore.</td>
</tr>
<tr>
<td>SWITCH_OPEN_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of ignored still-open buffer switches (resolved races).</td>
</tr>
</tbody>
</table>

Additional Information

The current configuration of in-memory log buffers is shown in the BUFFER_SIZE and BUFFER_COUNT columns. This defines how much log information can be collected transiently in memory, before the log queue becomes full.
Counters for buffer switches indicate performance of the in-memory log buffers. Normally, buffer switching happens without any waits. In the case of buffer full, however, a wait is necessary. Then, SWITCH_WAIT_COUNT is incremented, otherwise SWITCH_NOWAIT_COUNT is incremented. If the wait ratio is higher than one percent, this indicates a possible misconfiguration of the system. In this case:

- Check if regular peaks exceed current log buffer configuration and if so, increase log buffer size and/or count
- Check if the I/O subsystem is performing poorly (see the M_VOLUME_IO_TOTAL_STATISTICS system view).

Due to the lock-free nature of the algorithm used, some race conditions can happen. These are properly detected and resolved. Additionally, a count of such races is recorded in SWITCH_OPEN_COUNT. Normally, the ratio of races to buffer switches should also be under one percent even for high workloads.

This view has a resettable counterpart; you can see the values since the last reset in the M_LOG_BUFFERS_RESET system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_LOG_BUFFERS_RESET;`.

### Related Information

- M_LOG_BUFFERS_RESET System View [page 1996]
- M_VOLUME_IO_TOTAL_STATISTICS System View [page 2279]
- M_LOG_PARTITIONS System View [page 1997]
- M_LOG_SEGMENTS System View [page 2003]
- M_VOLUME_IO_TOTAL_STATISTICS System View [page 2279]
- Change the I/O Buffer Size
- Savepoints and Redo Logs

### 6.2.170 M_LOG_BUFFERS_RESET System View

Provides log buffer statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_LOG_BUFFERS. Refer to M_LOG_BUFFERS for information about the structure and use of this view.

In addition to the members mentioned in M_LOG_BUFFERS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

### Related Information

- M_LOG_BUFFERS System View [page 1995]
# 6.2.171 M_LOG_PARTITIONS System View

Provides log partition statistics.

## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>PARTITION_ID</td>
<td>BIGINT</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ble and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is in progress.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the log partition root path.</td>
</tr>
<tr>
<td>LAST_BUFFER_PREPARE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the last log buffer at prepare time (actual log data size).</td>
</tr>
<tr>
<td>MAX_BUFFER_PREPARE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the largest log buffer in bytes at prepare time (actual log data size).</td>
</tr>
<tr>
<td>MIN_BUFFER_PREPARE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the smallest log buffer at prepare time (actual log data size).</td>
</tr>
<tr>
<td>SUM_BUFFER_PREPARE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size in bytes of the log buffer at prepare time (actual log data size).</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AVG_BUFFER_PREPARE_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size in bytes of the log buffer at prepare time (actual log data size).</td>
</tr>
<tr>
<td>LAST_BUFFER_OVERHEAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the last log buffer alignment overhead at I/O time.</td>
</tr>
<tr>
<td>MAX_BUFFER_OVERHEAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest size in bytes of the log buffer alignment overhead at I/O time.</td>
</tr>
<tr>
<td>MIN_BUFFER_OVERHEAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the smallest size in bytes of the log buffer alignment overhead at I/O time.</td>
</tr>
<tr>
<td>SUM_BUFFER_OVERHEAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size in bytes of the log buffer alignment overhead at I/O time (total).</td>
</tr>
<tr>
<td>AVG_BUFFER_OVERHEAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size in bytes of the log buffer alignment overhead at I/O time.</td>
</tr>
<tr>
<td>LAST_BUFFER_IO_SIZE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the last log buffer at I/O time (actual data plus alignment overhead).</td>
</tr>
<tr>
<td>MAX_BUFFER_IO_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest size in bytes of the log buffer at I/O time (actual data plus alignment overhead).</td>
</tr>
<tr>
<td>MIN_BUFFER_IO_SIZE</td>
<td>BIGINT</td>
<td>Displays the smallest size in bytes of the log buffer at I/O time (actual data plus alignment overhead).</td>
</tr>
<tr>
<td>SUM_BUFFER_IO_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size in bytes of log buffer at I/O time (actual data plus alignment overhead).</td>
</tr>
<tr>
<td>AVG_BUFFER_IO_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size in bytes of log buffer at I/O time (actual data plus alignment overhead).</td>
</tr>
<tr>
<td>LAST_GROUP_COMMIT_FREQUENCY</td>
<td>BIGINT</td>
<td>Displays the count of the last group commit frequency (callback count per buffer with sync callback).</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_GROUP_COMMIT_FREQUENCY</td>
<td>BIGINT</td>
<td>Displays the count of the largest group commit frequency (callback count per buffer with sync callback).</td>
</tr>
<tr>
<td>MIN_GROUP_COMMIT_FREQUENCY</td>
<td>BIGINT</td>
<td>Displays the count of the smallest group commit frequency (callback count per buffer with sync callback).</td>
</tr>
<tr>
<td>SUM_GROUP_COMMIT_FREQUENCY</td>
<td>BIGINT</td>
<td>Displays the total count of the group commit frequency (callback count per buffer with sync callback).</td>
</tr>
<tr>
<td>AVG_GROUP_COMMIT_FREQUENCY</td>
<td>BIGINT</td>
<td>Displays the average count of the group commit frequency (callback count per buffer with sync callback).</td>
</tr>
<tr>
<td>LAST_CALLBACK_TIME</td>
<td>BIGINT</td>
<td>Displays the last callback time per buffer with sync callback in microseconds.</td>
</tr>
<tr>
<td>MAX_CALLBACK_TIME</td>
<td>BIGINT</td>
<td>Displays the longest callback time per buffer with sync callback in microseconds.</td>
</tr>
<tr>
<td>MIN_CALLBACK_TIME</td>
<td>BIGINT</td>
<td>Displays the shortest callback time per buffer with sync callback in microseconds.</td>
</tr>
<tr>
<td>SUM_CALLBACK_TIME</td>
<td>BIGINT</td>
<td>Displays the total callback time per buffer with sync callback in microseconds.</td>
</tr>
<tr>
<td>AVG_CALLBACK_TIME</td>
<td>BIGINT</td>
<td>Displays the average callback time per buffer with sync callback in microseconds.</td>
</tr>
<tr>
<td>PREPARED_BUFFERS</td>
<td>BIGINT</td>
<td>Displays the number of buffers prepared for I/O.</td>
</tr>
<tr>
<td>WRITTEN_BUFFERS</td>
<td>BIGINT</td>
<td>Displays the number of completed and written buffers.</td>
</tr>
<tr>
<td>WRITTEN_BUFFERS_OOO</td>
<td>BIGINT</td>
<td>Displays the number of buffers written out-of-order.</td>
</tr>
<tr>
<td>NEW_SEGMENT_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of new log segment requests.</td>
</tr>
<tr>
<td>FREE_SEGMENTS</td>
<td>BIGINT</td>
<td>Displays the number of log segments currently free.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IN_BACKUP_SEGMENTS</td>
<td>BIGINT</td>
<td>Displays the number of current in-backup log segments.</td>
</tr>
<tr>
<td>IN_BACKUP_TRUNCATED_SEGMENTS</td>
<td>BIGINT</td>
<td>Displays the number of current in-backup truncated log segments.</td>
</tr>
<tr>
<td>BACKED_UP_SEGMENTS</td>
<td>BIGINT</td>
<td>Displays the number of log segments backed up so far.</td>
</tr>
<tr>
<td>TOTAL_SEGMENTS</td>
<td>BIGINT</td>
<td>Displays the total number of log segment count in the partition.</td>
</tr>
<tr>
<td>RECOVERY_SEGMENTS_IN_LOAD</td>
<td>BIGINT</td>
<td>Displays the number of in-load segments during recovery.</td>
</tr>
<tr>
<td>RECOVERY_SEGMENTS_WAITING_FOR_LOAD</td>
<td>BIGINT</td>
<td>Displays the number of segments waiting for load during recovery.</td>
</tr>
<tr>
<td>RECOVERY_SEGMENTS_IN_PROCESS</td>
<td>BIGINT</td>
<td>Displays the number of segments in process during recovery.</td>
</tr>
<tr>
<td>RECOVERY_SEGMENTS_PROCESSED</td>
<td>BIGINT</td>
<td>Count of segments processed during recovery.</td>
</tr>
<tr>
<td>COMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of commits.</td>
</tr>
<tr>
<td>LAST_COMMIT_IO_LATENCY</td>
<td>BIGINT</td>
<td>Displays the last time in microseconds needed to synchronize the flush of commit log.</td>
</tr>
<tr>
<td>MAX_COMMIT_IO_LATENCY</td>
<td>BIGINT</td>
<td>Displays the longest time in microseconds needed to synchronize the flush of commit log entries.</td>
</tr>
<tr>
<td>MIN_COMMIT_IO_LATENCY</td>
<td>BIGINT</td>
<td>Displays the shortest time in microseconds needed in synchronize the flush of commit log entries.</td>
</tr>
<tr>
<td>SUM_COMMIT_IO_LATENCY</td>
<td>BIGINT</td>
<td>Displays the total time in microseconds needed to synchronize the flush of commit log entries.</td>
</tr>
<tr>
<td>AVG_COMMIT_IO_LATENCY</td>
<td>BIGINT</td>
<td>Displays the average time in microseconds needed to synchronize the flush of commit log entries.</td>
</tr>
</tbody>
</table>
**Additional Information**

This view collects various performance statistics for each log partition. The collected statistics can be used to optimize workload.

This view has a resettable counterpart; you can see the values since the last reset in the `M_LOG_PARTITIONS_RESET` system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_LOG_PARTITIONS_RESET;
```

**Related Information**

- `M_LOG_PARTITIONS_RESET` System View [page 2001]
- `M_LOG_BUFFERS` System View [page 1995]
- `M_LOG_SEGMENTS` System View [page 2003]
- `M_VOLUME_IO_TOTAL_STATISTICS` System View [page 2279]

**6.2.172 M_LOG_PARTITIONS_RESET System View**

Provides log partition statistics since the last reset.

This view contains values accumulated since the last reset of the main view `M_LOG_PARTITIONS`. Refer to `M_LOG_PARTITIONS` for information about the structure and use of this view.

In addition to the members mentioned in `M_LOG_PARTITIONS`, this view also contains a timestamp field, `RESET_TIME`, which indicates the last time the data was reset.

**Related Information**

- `M_LOG_PARTITIONS` System View [page 1997]
6.2.173  M_LOG_REPLAY_QUEUE_STATISTICS System View

Provides information about log replay queue statistics.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the name of the host.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>LOG_REPLAY_QUEUE_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the log replay queue.</td>
</tr>
<tr>
<td>LOG_RECORD_TYPE</td>
<td>VARCHAR(40)</td>
<td>Displays the type of redo log record.</td>
</tr>
<tr>
<td>TOTAL_LOG_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of replayed log entries.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the accumulated execute time to replay log entries in microseconds.</td>
</tr>
<tr>
<td>TOTAL_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the accumulated local wait time during replay of log entries in microseconds.</td>
</tr>
<tr>
<td>TOTAL_LOG_RECORD_SIZE</td>
<td>BIGINT</td>
<td>Displays the accumulated log size of replayed log entries.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_LOG_REPLAY_QUEUE_STATISTICS_RESET System View [page 2003]
- HOST_SERVICE_REPLICATION View (Embedded Statistics Service)
- Replaying a Workload
6.2.174 M_LOG_REPLAY_QUEUE_STATISTICS_RESET System View

Provides information about log replay queue statistics.

This view contains values accumulated since the last reset of the main view M_LOG_REPLAY_QUEUE_STATISTICS. Refer to M_LOG_REPLAY_QUEUE_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_LOG_REPLAY_QUEUE_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_LOG_REPLAY_QUEUE_STATISTICS System View [page 2002]

6.2.175 M_LOG_SEGMENTS System View

Provides log segment statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PARTITION_ID</td>
<td>BIGINT</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original</td>
</tr>
<tr>
<td></td>
<td></td>
<td>table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in progress.</td>
</tr>
<tr>
<td>SEGMENT_ID</td>
<td>BIGINT</td>
<td>Displays the log segment ID within the partition.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(512)</td>
<td>Displays the log segment file name.</td>
</tr>
<tr>
<td>FILE_OFFSET</td>
<td>BIGINT</td>
<td>Displays the start position of the log segment in the file.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STATE</td>
<td>VARCHAR(16)</td>
<td>Displays the log segment state:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Formatting: The log segment is being formatted and not yet used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Preallocated: The log segment has been preallocated, but never used.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Writing: The log segment is currently being written.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Closed: The log segment is closed, not backed up and is still required for</td>
</tr>
<tr>
<td></td>
<td></td>
<td>restart.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Truncated: The log segment is not required for restart, but has not been</td>
</tr>
<tr>
<td></td>
<td></td>
<td>backed up.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- BackedUp: The log segment has been backed up, but is still required for</td>
</tr>
<tr>
<td></td>
<td></td>
<td>restart.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- RetainedFree: The log segment has been backed up and is not required for</td>
</tr>
<tr>
<td></td>
<td></td>
<td>restart, but is required to resync the system replication sites.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Free: The log segment has been backed up, it is not required for restart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and can be reused.</td>
</tr>
<tr>
<td>MIN_POSITION</td>
<td>BIGINT</td>
<td>Displays the first position contained in the log segment.</td>
</tr>
<tr>
<td>MAX_POSITION</td>
<td>BIGINT</td>
<td>Displays the position behind the last log record in the log segment. This</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value is for closed log segments only.</td>
</tr>
<tr>
<td>HOLE_POSITION</td>
<td>BIGINT</td>
<td>Displays the start position of the log hole before the log segment. This</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value is equal to MIN_POSITION if there is no hole.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the used log segment size in bytes.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the total log segment size in bytes.</td>
</tr>
<tr>
<td>IN_BACKUP</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the flag for the log segment is in the backup: TRUE/FALSE</td>
</tr>
<tr>
<td>LAST_COMMIT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of the last commit in the log segment.</td>
</tr>
<tr>
<td>ENCRYPTION_KEY_HASH</td>
<td>VARCHAR(64)</td>
<td>Displays the hash of the key used for the log segment encryption.</td>
</tr>
</tbody>
</table>

**Additional Information description**

This view describes each allocated log segment and shows its current state and log position range that is currently contained in the segment.

This view has a resettable counterpart; you can see the values since the last reset in the M_LOG_SEGMENTS_RESET system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_LOG_SEGMENTS_RESET;`.

**Related Information**

- M_LOG_SEGMENTS_RESET System View [page 2006]
- M_LOG_BUFFERS System View [page 1995]
- M_LOG_PARTITIONS System View [page 1997]
- M_VOLUME_IO_TOTAL_STATISTICS System View [page 2279]

**6.2.176 M_LOG_SEGMENTS_RESET System View**

Provides log segment statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_LOG_SEGMENTS. Refer to M_LOG_SEGMENTS for information about the structure and use of this view.

In addition to the members mentioned in M_LOG_SEGMENTS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.
6.2.177  M_MEMORY System View

Deprecated. For service-specific information related to memory, use the M_SERVICE_MEMORY System View topic and for host-specific information related to memory, use the M_HOSTRESOURCE_UTILIZATION System View topic.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the type of memory.</td>
</tr>
<tr>
<td>VALUE</td>
<td>BIGINT</td>
<td>Displays the size of the memory in bytes.</td>
</tr>
</tbody>
</table>

Related Information

M_SERVICE_MEMORY System View [page 2113]
M_HOSTRESOURCE_UTILIZATION System View [page 1937]
Memory Management
Persistent Memory
Memory Sizing
SAP HANA Used Memory
View Memory Usage
6.2.178 M_MEMORY_OBJECTS System View

Returns memory object statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the object (statistic) type.</td>
</tr>
<tr>
<td>OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of objects currently in the memory object container.</td>
</tr>
<tr>
<td>OBJECT_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the objects that are currently in the memory object container in bytes.</td>
</tr>
<tr>
<td>NON_SWAPPABLE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the non-swappable objects that are currently in the memory object container in bytes.</td>
</tr>
<tr>
<td>SWAPPABLE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the swappable objects that are currently in the memory object container in bytes.</td>
</tr>
<tr>
<td>PUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of put objects.</td>
</tr>
<tr>
<td>PUT_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of put objects in bytes.</td>
</tr>
<tr>
<td>MOVE_IN_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of objects moved in from a different statistic.</td>
</tr>
<tr>
<td>MOVE_IN_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of objects moved in from a different statistic in bytes.</td>
</tr>
<tr>
<td>MOVE_OUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of objects moved out to a different statistic.</td>
</tr>
<tr>
<td>MOVE_OUT_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of objects moved out to a different statistic in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EVICT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of evicted objects.</td>
</tr>
<tr>
<td>EVICT_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of evicted objects in bytes.</td>
</tr>
<tr>
<td>TEMP_EVICT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of temp-evicted objects.</td>
</tr>
<tr>
<td>TEMP_EVICT_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of temp-evicted objects in bytes.</td>
</tr>
<tr>
<td>RESIZE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of resizes of objects.</td>
</tr>
<tr>
<td>RESIZE_DELTA_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size delta changed by the resize of the objects in bytes.</td>
</tr>
<tr>
<td>SHRINK_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of shrunken objects.</td>
</tr>
<tr>
<td>SHRINK_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of shrunken objects in bytes.</td>
</tr>
<tr>
<td>RETENTION_PERIOD_SHRINK_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of shrunken objects due to an unused retention per-</td>
</tr>
<tr>
<td>RETENTION_PERIOD_SHRINK_SIZE</td>
<td>BIGINT</td>
<td>iod.</td>
</tr>
<tr>
<td>PAGE_LOADABLE_COLUMNS_LIMIT_SHRINK_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of shrunken objects due to a paged attribute limit violation.</td>
</tr>
<tr>
<td>PAGE_LOADABLE_COLUMNS_LIMIT_SHRINK_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of shrunken objects due to paged attribute limit violation in bytes.</td>
</tr>
<tr>
<td>FAILED_SHRINK_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of the objects that the shrink failed to remove.</td>
</tr>
<tr>
<td>FAILED_SHRINK_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of the objects that the shrink failed to remove in bytes.</td>
</tr>
</tbody>
</table>
Additional Information

This view provides information about the number and size of resources in the resource container and about the throughput of the resource container. One row in this view represents one resource type (the type is specified by the resource statistic). The resource statistics are kept in a tree data structure and this view represents this tree in a human-readable form. Each value (except for HOST, PORT, VOLUME_ID, and STATISTICS_NAME) is the AGGREGATED value of the subtree (including the current node).

This view has a resettable counterpart; you can see the values since the last reset in the M_MEMORY_OBJECTS_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_MEMORY_OBJECTS_RESET;
```

Related Information

M_MEMORY_OBJECTS_RESET System View [page 2010]
M_MEMORY_OBJECT_DISPOSITIONS System View [page 2011]
M_MEMORY System View [page 2007]
Managing Memory by Object Usage

6.2.179 M_MEMORY_OBJECTS_RESET System View

Provides values accumulated since the last reset of the main view M_MEMORY_OBJECTS.

In addition to the members mentioned in M_MEMORY_OBJECTS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Refer to M_MEMORY_OBJECTS for information about the structure and use of this view.

Related Information

M_MEMORY_OBJECTS System View [page 2008]
## 6.2.180 M_MEMORY_OBJECT_DISPOSITIONS System View

Displays the disposition-specific memory object statistics.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the object statistic type.</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>VARCHAR(128)</td>
<td>Displays the allocator category, the corresponding allocator, and some of its sub-allocators that were used to allocate the memory objects.</td>
</tr>
<tr>
<td>TEMPORARY_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of temporary objects.</td>
</tr>
<tr>
<td>PAGE_LOADABLE_COLUMNS_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of page loadable column objects.</td>
</tr>
<tr>
<td>EARLY_UNLOAD_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of early unload objects.</td>
</tr>
<tr>
<td>INTERNAL_SHORT_TERM_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of internal short term objects.</td>
</tr>
<tr>
<td>SHORT_TERM_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of short term objects.</td>
</tr>
<tr>
<td>MID_TERM_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of mid term objects.</td>
</tr>
<tr>
<td>LONG_TERM_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of long term objects.</td>
</tr>
<tr>
<td>NON_SWAPPABLE_OBJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of non-swappable objects.</td>
</tr>
<tr>
<td>TEMPORAY_OBJECT_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the temporary objects in bytes.</td>
</tr>
</tbody>
</table>
### Additional Information

The number and the size of the resources in the resource container are shown depending on their specific disposition (whether the memory objects are short, mid, long term, or non-swappable). For each type of resource that is specified by the resource statistic, which is currently in the resource container, one row is added to this view. The tree structure of the resource statistics is not considered here, therefore the values are not aggregated.

Reading this view may take some time as the entire resource container must be traversed to generate this view.

### Related Information

- [M_MEMORY_OBJECTS System View](#) [page 2008]
- [M_MEMORY_OBJECTS_RESET System View](#) [page 2010]
- [M_MEMORY System View](#) [page 2007]
- Managing Memory by Object Usage
6.2.181 M_MEMORY_PROFILER System View

Provides memory profiler information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the file name.</td>
</tr>
<tr>
<td>PROFILE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the profile name.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>SAMPLING_INTERVAL</td>
<td>INTEGER</td>
<td>Displays the sampling interval in milliseconds.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time.</td>
</tr>
<tr>
<td>STOP_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the stop time.</td>
</tr>
<tr>
<td>REMAINING_SECONDS</td>
<td>INTEGER</td>
<td>Displays the remaining seconds until an automatic stop.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the status: STARTED/STOPPED.</td>
</tr>
<tr>
<td>HAS_CALLSTACKS</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the profiler has callstacks recorded: TRUE/FALSE.</td>
</tr>
<tr>
<td>FILE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the file in bytes.</td>
</tr>
</tbody>
</table>

Related Information

M_MEMORY System View [page 2007]
ALTER SYSTEM (START | STOP | SAVE | CLEAR) KERNEL PROFILER Statement (System Management) [page 571]
Kernel Profiler
### 6.2.182 M_MEMORY_RECLAIM_STATISTICS System View

Provides statistics for reclaiming memory (for example, defragmentation, unloading of memory objects, and so on).

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>STATISTICS_ID</td>
<td>BIGINT</td>
<td>Displays the statistics object unique ID.</td>
</tr>
<tr>
<td>MEMORY_RECLAIM_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of memory re-claims.</td>
</tr>
<tr>
<td>LAST_MEMORY_RECLAIM_DURATION</td>
<td>BIGINT</td>
<td>Displays the time needed in milliseconds to reclaim memory for the last allocation.</td>
</tr>
<tr>
<td>MAX_MEMORY_RECLAIM_DURATION</td>
<td>BIGINT</td>
<td>Displays the longest time needed in milliseconds to reclaim memory for the allocation.</td>
</tr>
<tr>
<td>MIN_MEMORY_RECLAIM_DURATION</td>
<td>BIGINT</td>
<td>Displays the shortest time needed in milliseconds to reclaim memory for the allocation.</td>
</tr>
<tr>
<td>SUM_MEMORY_RECLAIM_DURATION</td>
<td>BIGINT</td>
<td>Displays the total time needed in milliseconds to reclaim memory for all allocations.</td>
</tr>
<tr>
<td>AVG_MEMORY_RECLAIM_DURATION</td>
<td>BIGINT</td>
<td>Displays the average time needed in milliseconds to reclaim memory for the allocation.</td>
</tr>
<tr>
<td>LAST_SYNCHRONIZATION_DURATION</td>
<td>BIGINT</td>
<td>Displays the last wait time in milliseconds while other thread/process re-claims memory.</td>
</tr>
<tr>
<td>MAX_SYNCHRONIZATION_DURATION</td>
<td>BIGINT</td>
<td>Displays the longest wait time in milliseconds while other thread/process re-claims memory.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MIN_SYNCHRONIZATION_DURATION</td>
<td>BIGINT</td>
<td>Displays the shortest wait time in milliseconds while other thread/process re-claims memory.</td>
</tr>
<tr>
<td>SUM_SYNCHRONIZATION_DURATION</td>
<td>BIGINT</td>
<td>Displays the total wait time in milliseconds while other thread/process re-claims memory.</td>
</tr>
<tr>
<td>AVG_SYNCHRONIZATION_DURATION</td>
<td>BIGINT</td>
<td>Displays the average wait time in milliseconds while other thread/process re-claims memory.</td>
</tr>
<tr>
<td>LAST_DEFRAGMENTATION_DURATION</td>
<td>BIGINT</td>
<td>Displays the duration in milliseconds of the last defragmentation.</td>
</tr>
<tr>
<td>MAX_DEFRAGMENTATION_DURATION</td>
<td>BIGINT</td>
<td>Displays the longest duration in milliseconds of defragmentation.</td>
</tr>
<tr>
<td>MIN_DEFRAGMENTATION_DURATION</td>
<td>BIGINT</td>
<td>Displays the shortest duration in milliseconds of defragmentation.</td>
</tr>
<tr>
<td>SUM_DEFRAGMENTATION_DURATION</td>
<td>BIGINT</td>
<td>Displays the total duration in milliseconds of defragmentation.</td>
</tr>
<tr>
<td>AVG_DEFRAGMENTATION_DURATION</td>
<td>BIGINT</td>
<td>Displays the average duration in milliseconds of defragmentation.</td>
</tr>
<tr>
<td>LAST_DEFRAGMENTATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the size in bytes freed by the last defragmentation.</td>
</tr>
<tr>
<td>MAX_DEFRAGMENTATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest size in bytes freed by defragmentation.</td>
</tr>
<tr>
<td>MIN_DEFRAGMENTATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the smallest size in bytes freed by defragmentation.</td>
</tr>
<tr>
<td>SUM_DEFRAGMENTATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size freed in bytes by defragmentation.</td>
</tr>
<tr>
<td>AVG_DEFRAGMENTATION_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size in bytes freed by defragmentation.</td>
</tr>
<tr>
<td>LAST_MEMORY_OBJECT_UNLOAD_DURATION</td>
<td>BIGINT</td>
<td>Displays the duration in milliseconds of the last unload. Also see view M_MEMORY_OBJECTS.</td>
</tr>
<tr>
<td>MAX_MEMORY_OBJECT_UNLOAD_DURATION</td>
<td>BIGINT</td>
<td>Displays the longest duration in milliseconds of all unloads. Also see view M_MEMORY_OBJECTS.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MIN_MEMORY_OBJECT_UNLOAD_DURATION</td>
<td>BIGINT</td>
<td>Displays the shortest duration in milliseconds of all unloads. Also see view M_MEMORY_OBJECTS.</td>
</tr>
<tr>
<td>SUM_MEMORY_OBJECT_UNLOAD_DURATION</td>
<td>BIGINT</td>
<td>Displays the total duration in milliseconds of all unloads. Also see view M_MEMORY_OBJECTS.</td>
</tr>
<tr>
<td>AVG_MEMORY_OBJECT_UNLOAD_DURATION</td>
<td>BIGINT</td>
<td>Displays the average duration in milliseconds of all unloads. Also see view M_MEMORY_OBJECTS.</td>
</tr>
<tr>
<td>LAST_MEMORY_OBJECT_UNLOAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the size in bytes freed by the last unloading of memory objects. Also see view M_MEMORY_OBJECTS.</td>
</tr>
<tr>
<td>MAX_MEMORY_OBJECT_UNLOAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the largest size in bytes freed by unloading memory objects. Also see view M_MEMORY_OBJECTS.</td>
</tr>
<tr>
<td>MIN_MEMORY_OBJECT_UNLOAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the smallest size in bytes freed by unloading memory objects. Also see view M_MEMORY_OBJECTS.</td>
</tr>
<tr>
<td>SUM_MEMORY_OBJECT_UNLOAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size in bytes freed by unloading memory objects. Also see view M_MEMORY_OBJECTS.</td>
</tr>
<tr>
<td>AVG_MEMORY_OBJECT_UNLOAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size in bytes freed by unloading memory objects. Also see view M_MEMORY_OBJECTS.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view has a resettable counterpart; you can see the values since the last reset in the M_MEMORY_RECLAIM_STATISTICS_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_MEMORY_RECLAIM_STATISTICS_RESET;
```

**Related Information**

M_MEMORY_RECLAIM_STATISTICS_RESET System View [page 2017]
Memory Management
6.2.183 M_MEMORY_RECLAIM_STATISTICS_RESET System View

Provides statistics for reclaiming memory (for example, defragmentation, unloading of memory objects, and so on) since the last reset.

This view contains values accumulated since the last reset of the main view M_MEMORY_RECLAIM_STATISTICS. Refer to M_MEMORY_RECLAIM_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_MEMORY_RECLAIM_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_MEMORY_RECLAIM_STATISTICS System View [page 2014]

6.2.184 M_MERGED_TRACES System View

Contains the merged content of the server trace files for all of the SAP HANA services.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the database user.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction ID.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the update transaction ID.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>BIGINT</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>THREAD_ID</td>
<td>BIGINT</td>
<td>Displays the ID of thread that wrote trace entry.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the time when trace entry was written.</td>
</tr>
<tr>
<td>TRACE_LEVEL</td>
<td>VARCHAR(16)</td>
<td>Displays the trace level.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VARCHAR(16)</td>
<td>Displays the trace component.</td>
</tr>
<tr>
<td>SOURCE_FILE_NAME</td>
<td>VARCHAR(512)</td>
<td>Displays the source file which contains the trace.</td>
</tr>
<tr>
<td>SOURCE_FILE_LINE</td>
<td>BIGINT</td>
<td>Displays the source file line.</td>
</tr>
<tr>
<td>TRACE_TEXT</td>
<td>NCLOB</td>
<td>Displays the traced text.</td>
</tr>
<tr>
<td>TRACE_FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the name of the trace file containing the trace entry.</td>
</tr>
<tr>
<td>TRACE_FILE_LINE</td>
<td>INTEGER</td>
<td>Displays the trace file line.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application name.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays the application source.</td>
</tr>
<tr>
<td>PASSPORT_ROOT_CONTEXT_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the SAP EPP passport GUID identifying the source of request.</td>
</tr>
<tr>
<td>PASSPORT_TRANSACTION_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the SAP EPP passport GUID identifying the business transaction.</td>
</tr>
<tr>
<td>PASSPORT_CONNECTION_ID</td>
<td>VARCHAR(32)</td>
<td>Displays the SAP EPP passport GUID identifying the connection.</td>
</tr>
<tr>
<td>PASSPORT_CONNECTION_COUNTER</td>
<td>BIGINT</td>
<td>Displays the SAP EPP passport connection counter.</td>
</tr>
<tr>
<td>PASSPORT_COMPONENT_NAME</td>
<td>NVARCHAR(32)</td>
<td>Displays the SAP EPP passport component name of the initial/root context.</td>
</tr>
<tr>
<td>PASSPORT_ACTION</td>
<td>NVARCHAR(40)</td>
<td>Displays the SAP EPP passport action.</td>
</tr>
</tbody>
</table>

**Related Information**

MERGE INTO Statement (Data Manipulation) [page 1060]
The Delta Merge Operation
Traces
6.2.185 M_METADATA_CACHE_STATISTICS System View

Provides information regarding the efficiency and use of the metadata cache.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port number.</td>
</tr>
<tr>
<td>CACHE_GROUP_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the category name of the cache group.</td>
</tr>
<tr>
<td>ORIGIN_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the original database that the cache group belongs to.</td>
</tr>
<tr>
<td>ACCESS_COUNT</td>
<td>BIGINT</td>
<td>Displays the access count of the cache group.</td>
</tr>
<tr>
<td>HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the cache hit count of the cache group.</td>
</tr>
<tr>
<td>EMPTY_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the empty cache hit count of the cache group.</td>
</tr>
<tr>
<td>LATE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the duplicated metadata request count to the master node. This count only occurs in the race conditions of the thread.</td>
</tr>
<tr>
<td>NOT_FOUND_COUNT</td>
<td>BIGINT</td>
<td>Displays the empty result response count of the master node.</td>
</tr>
<tr>
<td>REVALIDATION_COUNT</td>
<td>BIGINT</td>
<td>Displays the revalidation count of old cache items.</td>
</tr>
<tr>
<td>MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the cache miss count of the cache group.</td>
</tr>
<tr>
<td>ENTRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of items in the cache group.</td>
</tr>
<tr>
<td>INVALID_ENTRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of invalid cache items. These items are set to invalid and removed during the next garbage collection.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VALID_ENTRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of valid cache items.</td>
</tr>
<tr>
<td>STALE_ENTRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of stale cache items. These are set as expired by de-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fault at the expiration time. These items can be revalidated later or re-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>moved at the next garbage collection.</td>
</tr>
<tr>
<td>USED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the memory occupied by the cache group in bytes.</td>
</tr>
</tbody>
</table>

**Related Information**

ALTER SYSTEM CLEAR CACHE Statement (System Management) [page 518]
ALTER SYSTEM REFRESH RESULT CACHE ENTRY Statement (System Management) [page 558]
ALTER SYSTEM CLEAR RESULT CACHE Statement (System Management) [page 522]
ALTER SYSTEM REMOVE RESULT CACHE ENTRY Statement (System Management) [page 561]
Generate Metadata to Configure Identity Provider Function Metadata Procedure Metadata

**6.2.186 M_MONITORS System View**

Provides available monitoring view information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the monitoring view.</td>
</tr>
</tbody>
</table>
### Additional Information

This view can be used in conjunction with the M_MONITOR_COLUMNS system view to get information about existing monitoring views.

### Related Information

- **M_MONITOR_COLUMNS System View** [page 2021]
- **ALTER SYSTEM RESET MONITORING VIEW Statement (System Management)** [page 565]
- Monitoring Views for the Backup Catalog
- System Monitor
- System Views for Monitoring Partitions
- Monitoring View Extensions for Column Store Paged Data Size

### 6.2.187 M_MONITOR_COLUMNS System View

All the columns in the monitoring views.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the monitoring view.</td>
</tr>
<tr>
<td>VIEW_COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the monitoring view column.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the column data type ID.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the column data type name.</td>
</tr>
</tbody>
</table>
### Additional Information

This view can be used in conjunction with the M_MONITORS system view to get information about existing monitoring views.

### Related Information

M_MONITORS System View [page 2020]
Monitoring View Extensions for Column Store Paged Data Size
6.2.188  M_MULTIDIMENSIONAL_STATEMENT_STATISTICS
System View

Displays all multidimensional statement statistics gathered since the server started. This information does not persist when you restart the server.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the statement hash.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application user.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
<tr>
<td>STATEMENT_TYPE</td>
<td>VARCHAR(3)</td>
<td>Displays the statement type as either INA or MDX.</td>
</tr>
<tr>
<td>LAST_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the last connection ID that executed the statement.</td>
</tr>
<tr>
<td>LAST_EXECUTION_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last execution.</td>
</tr>
<tr>
<td>LAST_METADATA_READ_DURA-</td>
<td>BIGINT</td>
<td>Displays the last metadata read duration.</td>
</tr>
<tr>
<td>TION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAST_QUERY_PREPARATION_</td>
<td>BIGINT</td>
<td>Displays the last query preparation duration.</td>
</tr>
<tr>
<td>DURATION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAST_PLAN_EXECUTION_DUR-</td>
<td>BIGINT</td>
<td>Displays the last plan execution duration.</td>
</tr>
<tr>
<td>ACTION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAST_POST_PROCESSING_DUR-</td>
<td>BIGINT</td>
<td>Displays the last post processing duration.</td>
</tr>
<tr>
<td>ACTION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAST_CUBE_PROCESSING_DUR-</td>
<td>BIGINT</td>
<td>Displays the last cube processing duration.</td>
</tr>
<tr>
<td>ACTION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LAST_PLAN_EXECUTION_RESULTSET_SIZE</td>
<td>BIGINT</td>
<td>Displays the last size of the intermediate result returned by the plan execution, in bytes.</td>
</tr>
<tr>
<td>LAST_RESULTSET_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the last result set, in bytes.</td>
</tr>
<tr>
<td>LAST_RESULTSET_CELL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of cells in the last result set.</td>
</tr>
<tr>
<td>LAST_HIERARCHY_COUNT</td>
<td>INTEGER</td>
<td>Displays the latest number of hierarchies.</td>
</tr>
<tr>
<td>LAST_CACHED_HIERARCHY_COUNT</td>
<td>INTEGER</td>
<td>Displays the last number of hierarchy cache hits.</td>
</tr>
<tr>
<td>LAST_CALCULATION_ENTITY_COUNT</td>
<td>INTEGER</td>
<td>Displays the last number of calculation entities.</td>
</tr>
<tr>
<td>LAST_CALCULATION_ENTITY_GROUP_COUNT</td>
<td>INTEGER</td>
<td>Displays the last number of calculation entity groups.</td>
</tr>
<tr>
<td>LAST_DRILL_DIMENSION_COUNT</td>
<td>INTEGER</td>
<td>Displays the last number of dimensions in the drill.</td>
</tr>
<tr>
<td>LAST_AGGREGATION_DIMENSION_COUNT</td>
<td>INTEGER</td>
<td>Displays the last number of aggregation dimensions.</td>
</tr>
<tr>
<td>LAST_SERIALIZED_CUBE_SIZE</td>
<td>BIGINT</td>
<td>Displays the last size of the serialized cube, in bytes.</td>
</tr>
<tr>
<td>LAST_REQUEST_QUEUE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of jobs waiting in the request queue.</td>
</tr>
<tr>
<td>LAST_EXECUTION_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the last execution status of the statement.</td>
</tr>
<tr>
<td>LAST_PERFORMANCE_DATA</td>
<td>NCLOB</td>
<td>Displays the last performance data.</td>
</tr>
<tr>
<td>EXECUTION_COUNT</td>
<td>INTEGER</td>
<td>Displays the execution count.</td>
</tr>
<tr>
<td>TOTAL_META_DATA_CACHE_HIT_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of data cache hits.</td>
</tr>
<tr>
<td>TOTAL_DATA_CACHE_HIT_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of data cached hits.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of the execution.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>AVG_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of the execution.</td>
</tr>
<tr>
<td>MIN_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of the execution.</td>
</tr>
<tr>
<td>MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of the execution.</td>
</tr>
<tr>
<td>MAX_CALLED_THREAD_COUNT</td>
<td>INTEGER</td>
<td>Displays the maximum thread count.</td>
</tr>
<tr>
<td>MAX_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum memory size in bytes.</td>
</tr>
</tbody>
</table>

**Related Information**

**Statements**
- Statement Performance
- System and Statistics Views
- The Statistics Service

**6.2.189 M_MUTEXES System View**

Provides information about single mutex (mutual exclusion) objects or groups of mutex objects.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>STATISTICS_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the statistics object name.</td>
</tr>
<tr>
<td>STATISTICS_ID</td>
<td>BIGINT</td>
<td>Displays the statistics object unique ID.</td>
</tr>
<tr>
<td>LOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of lock calls.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of blocking lock calls.</td>
</tr>
<tr>
<td>SPURIOUS_WAKEUP_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of spurious wake-ups (collisions on futex).</td>
</tr>
<tr>
<td>COLLISION_RATE</td>
<td>DOUBLE</td>
<td>Displays the collision rate percentage.</td>
</tr>
<tr>
<td>OWNER_ID</td>
<td>BIGINT</td>
<td>Displays the context ID of the owner context.</td>
</tr>
<tr>
<td>LAST_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the last time of the blocking lock calls in microseconds.</td>
</tr>
<tr>
<td>MAX_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of the blocking lock calls in microseconds.</td>
</tr>
<tr>
<td>MIN_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of the blocking lock calls in microseconds.</td>
</tr>
<tr>
<td>SUM_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of the blocking lock calls in microseconds.</td>
</tr>
<tr>
<td>AVG_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of the blocking lock calls in microseconds.</td>
</tr>
<tr>
<td>CREATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of mutex creations (for shared statistics only).</td>
</tr>
<tr>
<td>DESTROY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of mutex destructions (for shared statistics only).</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VARCHAR(32)</td>
<td>Displays the component.</td>
</tr>
</tbody>
</table>

**Additional Information**

M_MUTEXES does not contain information about all mutex objects. Information like LOCK_COUNT, WAIT_COUNT and WAIT_TIMES can be used to analyze performance bottlenecks. You can find possible deadlocks using the OWNER_ID column.

This view has a resettable counterpart; you can see the values since the last reset in the M_MUTEXES_RESET system view. To reset the view, execute the following statement: ALTER SYSTEM RESET MONITORING VIEW SYS.M_MUTEXES_RESET;.
6.2.190 M_MUTEXES_RESET System View

Provides mutex statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_MUTEXES. Refer to M_MUTEXES for information about the structure and use of this view.

In addition to the members mentioned in M_MUTEXES, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

6.2.191 M_MVCC_OVERVIEW System View

Provides an overview of the row-store Multiversion Concurrency Control (MVCC) manager.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of all versions in the host.</td>
</tr>
<tr>
<td>DATA_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of data versions per service.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>METADATA_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of metadata versions per service.</td>
</tr>
<tr>
<td>MIN_MVCC_SNAPSHOT_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the minimum MVCC timestamp which at least one transaction holds.</td>
</tr>
<tr>
<td>GLOBAL_MVCC_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the current global MVCC timestamp.</td>
</tr>
<tr>
<td>MIN_READ_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Indicates that all active transactions can see the changes of TRANSACTION IDs less than or equal to MIN_READ_TRANSACTION_ID.</td>
</tr>
<tr>
<td>MIN_WRITE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Indicates that all write transactions with a TRANSACTION ID less than or equal to MIN_WRITE_TRANSACTION_ID are closed.</td>
</tr>
<tr>
<td>NEXT_WRITE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the maximum TRANSACTION ID which is assigned to the next write transaction.</td>
</tr>
<tr>
<td>ACQUIRED_LOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of acquired records locks.</td>
</tr>
<tr>
<td>MAX_VERSIONS_PER_PARTITION</td>
<td>BIGINT</td>
<td>Displays the max number of versions per partition.</td>
</tr>
<tr>
<td>AVG_HASH_LINK_FOR_DATA_VERSIONS</td>
<td>REAL</td>
<td>Deprecated; do not use.</td>
</tr>
<tr>
<td>AVG_HASH_LINK_FOR_METADATA_VERSIONS</td>
<td>REAL</td>
<td>Deprecated; do not use.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_MVCC_SNAPSHOTS System View [page 2029]
- M_MVCC_TABLES System View [page 2030]
- M_SYSTEM_REPLICATION_MVCC_HISTOR Y System View [page 2196]
- CURRENT_MVCC_SNAPSHOT_TIMESTAMP Function (Datetime) [page 168]
- ALTER SYSTEM RECLAIM FLEXIBLE TABLES Statement (System Management) [page 552]
- Monitoring Multi-Host Health
- Key Performance Indicators
6.2.192 M_MVCC_SNAPSHOTS System View

Provides detailed snapshot information of the Multiversion Concurrency Control (MVCC) manager.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the associated transaction object ID.</td>
</tr>
<tr>
<td>MVCC_SNAPSHOT_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the MVCC timestamp of the snapshot.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>BIGINT</td>
<td>Displays the associated logical statement ID.</td>
</tr>
<tr>
<td>RELATED_TABLES</td>
<td>VARCHAR(1024)</td>
<td>Displays the associated table object IDs.</td>
</tr>
</tbody>
</table>

Related Information

- CURRENT_MVCC_SNAPSHOT_TIMESTAMP Function (Datetime) [page 168]
- M_MVCC_OVERVIEW System View [page 2027]
- M_MVCC_TABLES System View [page 2030]
- M_SNAPSHOTS System View [page 2134]
- M_TABLE_SNAPSHOTS System View [page 2226]
- M_TRANS_TOKENS System View [page 2253]
- BACKUP DATA CREATE SNAPSHOT Statement (Backup and Recovery) [page 706]
- BACKUP DATA DROP SNAPSHOT Statement (Backup and Recovery) [page 708]
- BACKUP DATA CLOSE SNAPSHOT Statement (Backup and Recovery) [page 704]
- Data Snapshots
6.2.193 M_MVCC_TABLES System View

Provides statistics for the row-store Multiversion Concurrency Control (MVCC) manager.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>VALUE</td>
<td>VARCHAR(128)</td>
<td>Displays the value.</td>
</tr>
</tbody>
</table>

Related Information

M_MVCC_OVERVIEW System View [page 2027]
M_MVCC_SNAPSHOTS System View [page 2029]
M_TRANS_TOKENS System View [page 2253]
Managing Tables

6.2.194 M_NUMA_NODES System View

Provides resource availability information on each NUMA node in the hardware topology, including inter-node distances and neighbor information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NUMA_NODE_ID</td>
<td>INTEGER</td>
<td>Displays the physical NUMA node ID as shown by the operating system.</td>
</tr>
<tr>
<td>NUMA_NODE_INDEX</td>
<td>INTEGER</td>
<td>Displays the logical NUMA node index. Indexes are in the range of 0 to MAX_NUMA_NODE_COUNT.</td>
</tr>
<tr>
<td>ACTIVE_LOGICAL_CORE_COUNT</td>
<td>INTEGER</td>
<td>Displays the total active logical core count in the NUMA node.</td>
</tr>
<tr>
<td>LOGICAL_CORE_IDS</td>
<td>VARCHAR(1024)</td>
<td>Displays the range-separated list of logical core IDs in the NUMA node, for example, (0-3,10-13).</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the total memory size, in bytes, present in the NUMA node.</td>
</tr>
<tr>
<td>NUMA_NODE_DISTANCES</td>
<td>VARCHAR(1024)</td>
<td>Displays the distances from this node to all other nodes. Indexes in this list are NUMA_NODE_INDEX and the number of entries are as many as MAX_NUMA_NODE_COUNT.</td>
</tr>
<tr>
<td>NEIGHBOUR_NUMA_NODE_IDS</td>
<td>VARCHAR(512)</td>
<td>Displays the physical NUMA node ID of the neighboring nodes.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_NUMA_RESOURCES System View [page 2032]
- Applying NUMA Location Preferences to Tables
- IMPORT Statement (Data Import Export) [page 1029]
- CREATE TABLE Statement (Data Definition) [page 825]
6.2.195 M_NUMA_RESOURCES System View

Provides information on overall resource availability for the system.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>MAX_NUMA_NODE_COUNT</td>
<td>INTEGER</td>
<td>Displays the maximum number of NUMA nodes.</td>
</tr>
<tr>
<td>MAX_LOGICAL_CORE_COUNT</td>
<td>INTEGER</td>
<td>Displays the maximum number of logical cores.</td>
</tr>
</tbody>
</table>

Related Information

M_NUMA_NODES System View [page 2030]
Applying NUMA Location Preferences to Tables
Registered Databases

6.2.196 M_OBJECT_LOCKS System View

Provides the status of currently acquired locks on objects with detailed information such as lock acquisition time and lock mode.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>LOCK_OWNER_TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction object ID owning the lock.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOCK_OWNER_UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the write transaction ID owning the lock.</td>
</tr>
<tr>
<td>ACQUIRED_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the lock acquisition time.</td>
</tr>
<tr>
<td>OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the object OID.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the locked partition, or 0 for a non-partition lock.</td>
</tr>
<tr>
<td>LOCK_MODE</td>
<td>VARCHAR(32)</td>
<td>Displays the lock mode: EXCLUSIVE/INTENTIONAL EXCLUSIVE.</td>
</tr>
</tbody>
</table>

**Related Information**

M_OBJECT_LOCK_STATISTICS System View [page 2033]
M_OBJECT_LOCK_STATISTICS_RESET System View [page 2035]
LOCK TABLE Statement (Transaction Management) [page 1056]
Blocked Transaction Monitoring

**6.2.197 M_OBJECT_LOCK_STATISTICS System View**

Provides lock contention statistics, including lock wait count, wait time, and failed count, for each object.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type: RECORD, TABLE, VIEW, SYNONYM, or SEQUENCE.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name that the object belongs to.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is in progress.</td>
</tr>
<tr>
<td>OBJECT_ID</td>
<td>BIGINT</td>
<td>Displays the object ID.</td>
</tr>
<tr>
<td>LOCK_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the lock type: OBJECT/RECORD.</td>
</tr>
<tr>
<td>LOCK_WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the lock wait count.</td>
</tr>
<tr>
<td>LOCK_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the lock wait time in microseconds.</td>
</tr>
<tr>
<td>LOCK_FAILED_COUNT</td>
<td>BIGINT</td>
<td>Displays the RECORD type lock failed count.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view has a resettable counterpart; you can see the values since the last reset in the M_OBJECT_LOCK_STATISTICS_RESET system view. To reset the view, execute the following statement: ALTER SYSTEM RESET MONITORING VIEW SYS.M_OBJECT_LOCK_STATISTICS_RESET;

**Related Information**

M_OBJECT_LOCK_STATISTICS_RESET System View [page 2035]
6.2.198  M_OBJECT_LOCK_STATISTICS_RESET System View

Provides lock contention statistics, including lock wait count, wait time, and failed count for each object since the last reset.

This view contains values accumulated since the last reset of the main view M_OBJECT_LOCK_STATISTICS. Refer to M_OBJECT_LOCK_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_OBJECT_LOCK_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_OBJECT_LOCK_STATISTICS System View [page 2033]

6.2.199  M_OUT_OF_MEMORY_EVENTS System View

Provides a list of the last 20 out-of-memory (OOM) events.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR (64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the OOM event.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR (20)</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR (32)</td>
<td>Displays the identifier for an SQL string.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the name of effective workload class</td>
</tr>
<tr>
<td>HEAP_MEMORY_CATEGORY</td>
<td>VARCHAR (128)</td>
<td>Displays the allocator name in case of a heap memory failure.</td>
</tr>
<tr>
<td>MEMORY_REQUEST_SIZE</td>
<td>BIGINT</td>
<td>Displays the size, in bytes, of the failed memory allocation.</td>
</tr>
<tr>
<td>MEMORY_USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the size, in bytes, currently allocated. This value depends on the EVENT_REASON.</td>
</tr>
<tr>
<td>EVENT_REASON</td>
<td>VARCHAR (32)</td>
<td>Displays the reason for the OOM event: GLOBAL ALLOCATION LIMIT, PROCESS ALLOCATION LIMIT, or STATEMENT MEMORY LIMIT.</td>
</tr>
<tr>
<td>MEMORY_LIMIT_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the limit in bytes. This value depends on the EVENT_REASON.</td>
</tr>
<tr>
<td>TRACEFILE_NAME</td>
<td>VARCHAR (256)</td>
<td>Displays the OOM trace file name.</td>
</tr>
</tbody>
</table>

**Related Information**

TEL_OUT_OF_MEMORY_EVENTS View (Embedded Statistics Service)
M_MEMORY System View [page 2007]
Memory Analysis
SAP HANA Used Memory

### 6.2.200  M_OUT_OF_MEMORY_EVENTS_RESET System View

Provides a list of the last out-of-memory (OOM) events since the last reset.

This view contains values accumulated since the last reset of the main view M_OUT_OF_MEMORY_EVENTS. Refer to M_OUT_OF_MEMORY_EVENTS for information about the structure and use of this view.

In addition to the members mentioned in M_OUT_OF_MEMORY_EVENTS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.
6.2.201  M_PAGEACCESS_STATISTICS System View

Provides PageAccess statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of pageaccess.</td>
</tr>
<tr>
<td>CHUNK_SIZE</td>
<td>INTEGER</td>
<td>Displays the chunk size in bytes.</td>
</tr>
<tr>
<td>ALLOCATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of allocations.</td>
</tr>
<tr>
<td>GET_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of gets.</td>
</tr>
<tr>
<td>LOAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of synchronous loads.</td>
</tr>
<tr>
<td>TRIGGER_LOAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of asynchronous loads.</td>
</tr>
<tr>
<td>DEALLOCATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deallocations.</td>
</tr>
</tbody>
</table>

Additional Information

This view contains information about pages accessed:

- TYPE specifies the pageaccess.
- The *_COUNT values are the number of the respective operations on the page access.
- CHUNK_SIZE is the number of pages accessed by one operation (CHUNK_SIZE is 1 except for RowStorePageAccess, where blocks of pages are accessed. For example, LOAD_COUNT=4 and CHUNK_SIZE=1024 means that 2*1024=4096 pages had been loaded). the corresponding Converter is accessed on allocation and deallocation.
This view has a resettable counterpart: you can see the values since the last reset in the M_PAGEACCESS_STATISTICS_RESET system view. To reset the view, execute the following statement: ALTER SYSTEM RESET MONITORING VIEW SYS.M_PAGEACCESS_STATISTICS_RESET;

Related Information

M_PAGEACCESS_STATISTICS_RESET System View [page 2038]
ALTER SYSTEM RESET MONITORING VIEW Statement (System Management) [page 565]
M_CONVERTER_STATISTICS System View [page 1805]
Reduce the Memory Footprint Using Page-Loadable Columns
Data Access

6.2.202 M_PAGEACCESS_STATISTICS_RESET System View

Provides the PageAccess statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_PAGEACCESS_STATISTICS. Refer to M_PAGEACCESS_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_PAGEACCESS_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_PAGEACCESS_STATISTICS System View [page 2037]
### 6.2.203 M_PASSWORD_POLICY System View

Defines effective password policy settings.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPERTY</td>
<td>VARCHAR(128)</td>
<td>Displays the configuration property. The following describes the contents of the individual lines of the monitor view (VALUE), as identified by PROPERTY:</td>
</tr>
<tr>
<td>force_first_password_change</td>
<td></td>
<td>Displays if a user is forced to change their administrator-given password before being allowed to work any further. This property is only applicable to users connecting using their SAP HANA database user name and password.</td>
</tr>
<tr>
<td>last_used_passwords</td>
<td></td>
<td>Displays the number of recently used passwords of a user, which they cannot reuse.</td>
</tr>
<tr>
<td>maximum_invalid_connect_attempts</td>
<td></td>
<td>Displays the maximum number of allowed invalid connect attempts before a user is locked out.</td>
</tr>
<tr>
<td>maximum_password_lifetime</td>
<td></td>
<td>Displays the number of days that a password stays valid.</td>
</tr>
<tr>
<td>maximum_unused_initial_password_lifetime</td>
<td></td>
<td>Displays the number of days that an unused administrator-given password stays valid.</td>
</tr>
</tbody>
</table>

**i Note**

In SAP HANA 1.0 SP 12 and earlier, the parameter name is maximum_unused_initial_password_lifetime (note the missing ‘i’ in ‘initial’). Use of the previous spelling is supported, but it is recommended that you update your...
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE</td>
<td>VARCHAR(128)</td>
<td>Displays the value.</td>
</tr>
</tbody>
</table>

### Related Information

Configure Password Policies  
Password Policy Configuration Options  
Configure a Password Policy for a User Group  
CREATE USERGROUP Statement (Access Control) [page 902]  
ALTER USERGROUP Statement (Access Control) [page 665]  
M_EFFECTIVE_PASSWORD_POLICY System View [page 1879]
6.2.204 M_PERFTRACE System View

Displays the state of the current performance trace. The performance trace provides detailed information about query execution.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the status of the performance trace: STOPPED, STARTED, or SAVING.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays when the performance trace started.</td>
</tr>
<tr>
<td>STOP_TIME</td>
<td>TIMESTAMP</td>
<td>Displays when the performance trace stopped.</td>
</tr>
<tr>
<td>FILE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the collected trace files in bytes. This value is only valid if tracing is stopped and not saved.</td>
</tr>
<tr>
<td>REMAINING_SECONDS</td>
<td>INTEGER</td>
<td>Displays the remaining number of seconds until tracing stops automatically.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the SQL user name filter.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user name filter.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application name filter.</td>
</tr>
<tr>
<td>PASSPORT_TRACELEVEL</td>
<td>VARCHAR(8)</td>
<td>Displays the passport filter: MEDIUM/HIGH.</td>
</tr>
<tr>
<td>PLAN_EXECUTION</td>
<td>VARCHAR(5)</td>
<td>Indicates whether plan execution details are recorded.</td>
</tr>
<tr>
<td>FUNCTION_PROFILER</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the function profiler details are recorded: TRUE/FALSE.</td>
</tr>
<tr>
<td>ROOT_STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the root statement hash filter.</td>
</tr>
</tbody>
</table>
Related Information

ALTER SYSTEM {START | STOP} PERFTRACE Statement (System Management) [page 578]
ALTER SYSTEM SAVE PERFTRACE Statement (System Management) [page 566]
Performance Trace
View Diagnostic Files in the SAP HANA Database Explorer
Performance Trace

6.2.205 M_PERSISTENCE_ENCRYPTION_KEYS System View

Provides information about encryption page keys.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VALID_FROM_SAVEPOINT</td>
<td>INTEGER</td>
<td>Displays the valid from savepoint version.</td>
</tr>
<tr>
<td>VALID_FROM_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the valid from timestamp, in microseconds, given in UTC.</td>
</tr>
<tr>
<td>IS_ENCRYPTED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the persistence encryption is active: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

M_PERSISTENCE_ENCRYPTION_STATUS System View [page 2043]
ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
SAP HANA Backup Encryption
Encryption Configuration
SQLScript Encryption
6.2.206  M_PERSISTENCE_ENCRYPTION_STATUS System View

Provides information about persistence encryption.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>ENCRYPTION_ACTIVE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the encryption is currently active for writing into persistence: TRUE/FALSE.</td>
</tr>
<tr>
<td>ENCRYPTION_ACTIVE_AFTER_NEXT_SAVEPOINT</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the encryption is active for writing into persistence after the next savepoint: TRUE/FALSE.</td>
</tr>
<tr>
<td>KEY_CHANGE_WITH_NEXT_SAVEPOINT</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the next savepoint activates a new key: TRUE/FALSE.</td>
</tr>
<tr>
<td>ROOT_KEY_CHANGE_WITH_NEXT_SAVEPOINT</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the next savepoint activates a new root key: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_LATEST_ROOT_KEY_VERSION</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the used root key version is the newest one: TRUE/FALSE.</td>
</tr>
<tr>
<td>USED_ROOT_KEY_VERSION</td>
<td>INTEGER</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>DATA_CONVERSION_ACTIVE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the conversion of data to the latest encryption status or key is active: TRUE/FALSE.</td>
</tr>
<tr>
<td>KEY_PAGE_ENTRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of entries within the key page.</td>
</tr>
<tr>
<td>USED_ROOT_KEY_HASH</td>
<td>VARCHAR(64)</td>
<td>Displays the hash of the root key that this service is using.</td>
</tr>
</tbody>
</table>
Related Information

M_PERSISTENCE_ENCRYPTION_KEYS System View [page 2042]
ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
SAP HANA Backup Encryption
Encryption Configuration
SQLScript Encryption

6.2.207 M_PERSISTENCE_MANAGERS System View

Provides persistence manager statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>MAX_TID</td>
<td>BIGINT</td>
<td>Displays the maximum known TID.</td>
</tr>
<tr>
<td>CCH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of normal consistent changes in terminated sessions.</td>
</tr>
<tr>
<td>MASS_CCH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of consistent changes for mass operations in terminated sessions.</td>
</tr>
<tr>
<td>LAST_CCH_TIME</td>
<td>BIGINT</td>
<td>Displays the last time spent in normal consistent changes in terminated sessions in microseconds.</td>
</tr>
<tr>
<td>MAX_CCH_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time spent in normal consistent changes in terminated sessions in microseconds.</td>
</tr>
<tr>
<td>MIN_CCH_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time spent in normal consistent changes in terminated sessions in microseconds.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SUM_CCH_TIME</td>
<td>BIGINT</td>
<td>Displays the total time spent in normal consistent changes in terminated sessions in microseconds.</td>
</tr>
<tr>
<td>AVG_CCH_TIME</td>
<td>BIGINT</td>
<td>Displays the average time spent in normal consistent changes in terminated sessions in microseconds.</td>
</tr>
<tr>
<td>LAST_MASS_CCH_TIME</td>
<td>BIGINT</td>
<td>Displays the last time spent in consistent changes for mass operations in terminated sessions in microseconds.</td>
</tr>
<tr>
<td>MAX_MASS_CCH_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time spent in consistent changes for mass operations in terminated sessions in microseconds.</td>
</tr>
<tr>
<td>MIN_MASS_CCH_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time spent in consistent changes for mass operations in terminated sessions in microseconds.</td>
</tr>
<tr>
<td>SUM_MASS_CCH_TIME</td>
<td>BIGINT</td>
<td>Displays the total time spent in consistent changes for mass operations in terminated sessions in microseconds.</td>
</tr>
<tr>
<td>AVG_MASS_CCH_TIME</td>
<td>BIGINT</td>
<td>Displays the average time spent in consistent changes for mass operations in terminated sessions in microseconds.</td>
</tr>
<tr>
<td>PREPARE_COMMIT_POS</td>
<td>BIGINT</td>
<td>Displays the newest known log position of the prepare commit. This value is for the slave only.</td>
</tr>
<tr>
<td>MASTER_COMMIT_POS</td>
<td>BIGINT</td>
<td>Displays the newest known log position of the commit record on the transaction master from the point of view of this node.</td>
</tr>
<tr>
<td>INDOUBT_WAITERS</td>
<td>BIGINT</td>
<td>Displays the number of in-doubt waiters for the currently-running COMMIT. This value is for the master only.</td>
</tr>
<tr>
<td>INDOUBT_RESTART_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of open in-doubt transactions before the restart. This value is for the master only.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
INDOUBT_ONLINE_COUNT | BIGINT | Displays the number of open in-doubt transactions since the last restart. This value is for the master only.
SAVEPOINT_CONFIG_FREQUENCY | BIGINT | Displays the configured savepoint frequency in seconds.
SAVEPOINT_ACTIVE_FREQUENCY | BIGINT | Displays the currently active savepoint frequency in seconds.
CHECKSUM_ALGORITHM | VARCHAR(16) | Displays the currently used checksum algorithm for the modified pages and log buffers.

### Additional Information

Persistence manager is the module responsible for low-level operations on the persistent data structures. This view shows various statistics counters, which are used to measure the performance of those operations.

This view has a resettable counterpart; you can see the values since the last reset in the `M_PERSISTENCE_MANAGERS_RESET` system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_PERSISTENCE_MANAGERS_RESET;
```

### Related Information

- M_PERSISTENCE_MANAGERS_RESET System View [page 2046]
- M_GARBAGE_COLLECTION_STATISTICS System View [page 1920]
- ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
- Persistent Memory

### 6.2.208 M_PERSISTENCE_MANAGERS_RESET System View

Provides persistence manager statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_PERSISTENCE_MANAGERS. Refer to M_PERSISTENCE_MANAGERS for information about the structure and use of this view.

In addition to the members mentioned in M_PERSISTENCE_MANAGERS, this view also contains a timestamp field, `RESET_TIME`, which indicates the last time the data was reset.
6.2.209  M_PERSISTENT_MEMORY_VOLUMES System View

Reports the capacity, usage and metadata of persistent memory volumes.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays an ID for the persistence volume.</td>
</tr>
<tr>
<td>NUMA_NODE_INDEX</td>
<td>SMALLINT</td>
<td>Displays the logical NUMA node index as in the M_NUMA_NODES view.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the absolute directory base path of persistent memory volume configured for NUMA node</td>
</tr>
<tr>
<td>FILESYSTEM_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the filesystem type, for example, ext3, ext4, xfs.</td>
</tr>
<tr>
<td>IS_DIRECT_ACCESS_SUPPORTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether Direct Access semantics are supported by the volume.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the total physical memory size, in bytes, of the persistent memory volume.</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the used physical memory size, in bytes, of persistent memory volume.</td>
</tr>
</tbody>
</table>
6.2.210 M_PERSISTENT_MEMORY_VOLUME_DATA_FILES System View

Reports metadata statistics about files created by SAP HANA services for data storage on the persistent memory volumes.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays an ID for the persistence volume.</td>
</tr>
<tr>
<td>NUMA_NODE_INDEX</td>
<td>SMALLINT</td>
<td>Displays the logical NUMA node index as in the M_NUMA_NODES view.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the file name of data block file located on persistent memory volume on Numa node of host (for example, 000000fe0000018a-00000051-000001c2-00000001_159.file block).</td>
</tr>
<tr>
<td>IS_MAPPED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether the data block file is memory mapped into virtual address space: TRUE or FALSE.</td>
</tr>
<tr>
<td>SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the data block file in bytes.</td>
</tr>
</tbody>
</table>
Related Information

M_PERSISTENT_MEMORY_VOLUMES System View [page 2047]
M_PERSISTENT_MEMORY_VOLUME_STATISTICS System View [page 2049]
M_PERSISTENT_MEMORY_VOLUME_STATISTICS_RESET System View [page 2051]
Persistent Memory
Persistent Data Storage in the SAP HANA Database
ALTER SYSTEM PERSISTENCE ENCRYPTION Statement (System Management) [page 545]
UNLOAD Statement (Data Manipulation) [page 1179]
M_CATALOG_MEMORY System View [page 1769]

6.2.211 M_PERSISTENT_MEMORY_VOLUME_STATISTICS
System View

Reports the statistics of physical lifecycle events of blocks managed by SAP HANA services on the persistent memory volumes.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays an ID for the persistence volume.</td>
</tr>
<tr>
<td>NUMA_NODE_INDEX</td>
<td>SMALLINT</td>
<td>Displays the logical NUMA node index as in the M_NUMA_NODES view.</td>
</tr>
<tr>
<td>TOTAL_ACTIVE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of active blocks from persistent memory volume.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This counter is computed as a sum of loaded or created blocks but excludes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>deleted blocks.</td>
</tr>
<tr>
<td>TOTAL_ACTIVE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size, in bytes, of active blocks from persistent memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>volume. This counter is computed as a sum of loaded or created blocks but</td>
</tr>
<tr>
<td></td>
<td></td>
<td>excludes deleted blocks.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TOTAL_CREATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of newly created (that is, temporary) blocks from the persistent memory volume.</td>
</tr>
<tr>
<td>TOTAL_CREATE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size, in bytes, of the newly created temporary blocks from the persistent memory volume.</td>
</tr>
<tr>
<td>TOTAL_COMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of the permanently persisted blocks from the persistent memory volume.</td>
</tr>
<tr>
<td>TOTAL_COMMIT_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size, in bytes, of the permanently persisted blocks from the persistent memory volume.</td>
</tr>
<tr>
<td>TOTAL_DELETE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of deleted blocks from the persistent memory volume.</td>
</tr>
<tr>
<td>TOTAL_DELETE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size, in bytes, of the deleted blocks from the persistent memory volume.</td>
</tr>
<tr>
<td>TOTAL_DESTROY_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of destroyed blocks (no refcount) from the persistent memory volume.</td>
</tr>
<tr>
<td>TOTAL_DESTROY_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size, in bytes, of the destroyed blocks (no refcount) from the persistent memory volume.</td>
</tr>
<tr>
<td>TOTAL_LOAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of loaded/initialized blocks from the persistent memory volume upon service restart.</td>
</tr>
<tr>
<td>TOTAL_LOAD_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size, in bytes, of loaded/initialized blocks from the persistent memory volume upon service restart.</td>
</tr>
<tr>
<td>TOTAL_MAPPED_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of blocks mapped into virtual memory from the persistent memory volume.</td>
</tr>
<tr>
<td>TOTAL_MAPPED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size, in bytes, of blocks mapped into virtual memory from the persistent memory volume.</td>
</tr>
</tbody>
</table>
M_PERSISTENT_MEMORY_VOLUME_STATISTICS_RESET System View

Reports the statistics of physical lifecycle events of blocks managed by SAP HANA services on the persistent memory volumes since the last reset.

This view contains values accumulated since the last reset of the main view M_PERSISTENCE_MEMORY_VOLUME_STATISTICS. Refer to M_PERSISTENCE_MEMORY_VOLUME_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_PERSISTENCE_MEMORY_VOLUME_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_PERSISTENT_MEMORY_VOLUME_STATISTICS System View [page 2049]
6.2.213  M_PLE_RUNTIME_OBJECTS System View

Lists all the internal cache objects created to support the planning sessions, with details about each one.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• LOOKUP DICTIONARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TIMETABLE DICTIONARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• FISCPTERTABLE DICTIONARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CONVERSTION DICTIONARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• FOX REFERENCE DATA DICTIONARY</td>
</tr>
<tr>
<td>OBJECT_SCOPE</td>
<td>VARCHAR(45)</td>
<td>Displays the object scope: SESSION/GLOBAL.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(5000)</td>
<td>Displays the object name. This name is a generated, unique name for the runtime object, derived from the corresponding session group name, session name, planning command name, and source columns.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the creation time.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size, in bytes.</td>
</tr>
<tr>
<td>ENTRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of entries in the runtime object.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SITE_ID</td>
<td>INTEGER</td>
<td>Displays the site ID of the secondary site. This value is -1 on a single instance.</td>
</tr>
</tbody>
</table>

**Related Information**

M_PLE_SESSIONS System View [page 2053]
6.2.214 M_PLE_SESSIONS System View

Lists all planning sessions on the system as well as their status and details.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>SESSION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the session name.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the session is valid: TRUE or FALSE.</td>
</tr>
<tr>
<td>SESSION_GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the session group name.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SITE_ID</td>
<td>INTEGER</td>
<td>Displays the site ID of the secondary site. This value is -1 on a single instance.</td>
</tr>
</tbody>
</table>

**Related Information**

M_PLE_RUNTIME_OBJECTS System View [page 2052]
Session Monitoring
### 6.2.215 M_PLUGIN_MANIFESTS System View

Provides information about installed plugins.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLUGIN_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the plugin name.</td>
</tr>
<tr>
<td>KEY</td>
<td>VARCHAR(64)</td>
<td>Displays the key.</td>
</tr>
<tr>
<td>VALUE</td>
<td>VARCHAR(256)</td>
<td>Displays the value.</td>
</tr>
</tbody>
</table>

**Related Information**

M_PLUGIN_STATUS System View [page 2054]
Configure the Default Build Plug-in Libraries Available to an SAP HDI Container
List The Plug-in Libraries That Can Be Configured for an SAP HDI Container
Configure a Custom Set of Build Plug-in Libraries Available to an SAP HDI Container
List All Currently Configured Build Plug-in Libraries Available to an SAP HDI Container

### 6.2.216 M_PLUGIN_STATUS System View

Provides status for plugins.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLUGIN_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the subdirectory, containing executables and plugins, where the files of an AFL area are located. There can be several areas per subdirectory.</td>
</tr>
<tr>
<td>ERROR_TEXT</td>
<td>NVARCHAR(1024)</td>
<td>Displays the error message in case a package or AFL cannot be registered.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AREA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of an area.</td>
</tr>
<tr>
<td>AREA_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the status of an area: REGISTRATION SUCCESSFUL, REGISTRATION FAILED, or REGISTRATION SKIPPED.</td>
</tr>
<tr>
<td>PACKAGE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of a package, which is an internal representation of an AFL. There can be several packages per area.</td>
</tr>
<tr>
<td>PACKAGE_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the status of a package: REGISTRATION SUCCESSFUL, REGISTRATION FAILED, or REGISTRATION SKIPPED.</td>
</tr>
</tbody>
</table>

**Related Information**

M_PLUGIN_MANIFESTS System View [page 2054]
- Configure the Default Build Plug-in Libraries Available to an SAP HDI Container
- List The Plug-in Libraries That Can Be Configured for an SAP HDI Container
- Configure a Custom Set of Build Plug-in Libraries Available to an SAP HDI Container
- List All Currently Configured Build Plug-in Libraries Available to an SAP HDI Container

**6.2.217 M_PREPARED_STATEMENTS System View**

Provides information about prepared statements.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the prepared statement ID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the MD5 hash value for the statement string.</td>
</tr>
<tr>
<td>START_MVCC_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the internal MVCC timestamp of the transaction start time.</td>
</tr>
<tr>
<td>COMPILED_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the compilation timestamp of the statement.</td>
</tr>
<tr>
<td>STATEMENT_STATUS</td>
<td>VARCHAR(128)</td>
<td>Displays the status of the SQL statement.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the SQL statement.</td>
</tr>
</tbody>
</table>
| ALLOCATED_MEMORY_SIZE    | BIGINT        | Displays the memory peak used for executing this statement. In case of dist-
|                          |               | tributed execution it is a sum of the local peak memories of multiple servers.|
| USED_MEMORY_SIZE         | BIGINT        | Deprecated. This value is the same as ALLOCATED_MEMORY_SIZE.                 |
| PLAN_ID                  | BIGINT        | Displays the logical plan ID.                                               |
| LAST_EXECUTED_TIME       | TIMESTAMP     | Displays the recently executed time of the statement. This timestamp is up-
|                          |               | dated when opening cursors and executing DML/DDL but not when fetching cur-
|                          |               | sor results or closing cursors.                                             |
| LAST_ACTION_TIME         | TIMESTAMP     | Displays the recently performed time of action against the statement. This
|                          |               | timestamp is updated when opening cursors, executing DML/DDL, fetching re-
<p>|                          |               | sults, and closing cursors.                                                 |
| RECOMPILE_COUNT          | BIGINT        | Displays the recompile count.                                               |
| EXECUTION_COUNT          | BIGINT        | Displays the count of executions.                                           |
| AVG_EXECUTION_TIME       | BIGINT        | Displays the average time of statement execution.                           |
| MAX_EXECUTION_TIME       | BIGINT        | Displays the maximum time of statement execution.                           |
| MIN_EXECUTION_TIME       | BIGINT        | Displays the minimum time of statement execution.                           |</p>
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of statement execution time.</td>
</tr>
<tr>
<td>AVG_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the average time of statement execution including communication time with clients.</td>
</tr>
<tr>
<td>MAX_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the maximum time of statement execution including communication time with clients.</td>
</tr>
<tr>
<td>MIN_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the minimum time of statement execution including communication time with clients.</td>
</tr>
<tr>
<td>TOTAL_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the sum of statement execution time including communication time with clients.</td>
</tr>
<tr>
<td>AVG_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the average memory size in bytes used during each execution.</td>
</tr>
<tr>
<td>MAX_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum memory size in bytes used during each execution.</td>
</tr>
<tr>
<td>MIN_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum memory size in bytes used during each execution.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the sum of memory size in bytes used during each execution.</td>
</tr>
<tr>
<td>AVG_LOCKWAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the average lock wait time for the statement.</td>
</tr>
<tr>
<td>MAX_LOCKWAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum lock wait time for the statement.</td>
</tr>
<tr>
<td>MIN_LOCKWAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum lock wait time for the statement.</td>
</tr>
<tr>
<td>TOTAL_LOCKWAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total lock wait count for the statement.</td>
</tr>
<tr>
<td>TOTAL_LOCKWAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the accumulated lock wait time for the statement.</td>
</tr>
<tr>
<td>AVG_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of statement preparation.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of statement preparation.</td>
</tr>
<tr>
<td>MIN_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of statement preparation.</td>
</tr>
<tr>
<td>TOTAL_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of statement preparation.</td>
</tr>
<tr>
<td>TOTAL_PREPARATION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total count of statement preparation.</td>
</tr>
<tr>
<td>HAS_HOLDABLE_CURSOR</td>
<td>VARCHAR(5)</td>
<td>Displays the holdable cursor existence.</td>
</tr>
<tr>
<td>CURSOR_TYPE</td>
<td>VARCHAR(18)</td>
<td>Displays the type of cursor.</td>
</tr>
<tr>
<td>PARENT_STATEMENT_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the parent prepared statement ID.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays the application source information.</td>
</tr>
<tr>
<td>STATEMENT_MEMORY_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective statement memory limit.</td>
</tr>
<tr>
<td>STATEMENT_THREAD_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective statement thread limit.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_MEMORY_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective total statement memory limit.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_THREAD_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective total statement thread limit.</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the effective workload class used for the execution.</td>
</tr>
</tbody>
</table>

**Additional Information**

Setting the value of the APPLICATION_SOURCE is only available via internal APIs of the SAP HANA database client interfaces (for more information see SAP Note 2873396).

**Related Information**

SAP Note 2873396
Execute SQL Statements
Execute SQL Statements in SAP HANA Studio
Use User-Defined SQL Statements for System Monitoring
Reduce the Complexity of SQL Statements

6.2.218  M_QUERY_CACHED_PLANS System View - Deprecated

Provides query cached plan information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>ABAP_VARCHAR_MODE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not ABAP VARCHAR mode is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_REMOTE</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not REMOTE is supported: TRUE/FALSE.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NLCOB</td>
<td>Displays the SQL statement.</td>
</tr>
<tr>
<td>PLAN_ID</td>
<td>BIGINT</td>
<td>Displays the logical plan ID.</td>
</tr>
<tr>
<td>REFERENCE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of statements using the plan.</td>
</tr>
<tr>
<td>USED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the current snapshot of the used memory size for executing this statement in bytes.</td>
</tr>
<tr>
<td>SHARING_TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the sharing type.</td>
</tr>
<tr>
<td>OWNER_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID of the owner.</td>
</tr>
</tbody>
</table>
Related Information

ALTER SYSTEM CLEAR SQL PLAN CACHE Statement (System Management) [page 522]
ALTER SYSTEM RECOMPILE SQL PLAN CACHE ENTRY Statement (System Management) [page 555]
ALTER SYSTEM (PIN | UNPIN) SQL PLAN CACHE ENTRY Statement (System Management) [page 548]
Monitoring SQL Performance with the SQL Plan Cache
Monitor and Analyze Statements with SQL Plan Cache

6.2.219 M_READWRITELOCKS System View

Provides read and write lock statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>STATISTICS_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the statistics object name.</td>
</tr>
<tr>
<td>STATISTICS_ID</td>
<td>BIGINT</td>
<td>Displays the statistics object unique ID.</td>
</tr>
<tr>
<td>OWNER_ID</td>
<td>BIGINT</td>
<td>Displays the context ID of the owner context (for exclusive/intent locks).</td>
</tr>
<tr>
<td>EXCLUSIVE_LOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of exclusive lock calls.</td>
</tr>
<tr>
<td>EXCLUSIVE_WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of blocking exclusive lock calls.</td>
</tr>
<tr>
<td>EXCLUSIVE_CAS_COLLISION_COUNT</td>
<td>BIGINT</td>
<td>Displays the collision count on atomic operations on exclusive locks.</td>
</tr>
<tr>
<td>EXCLUSIVE_COLLISION_RATE</td>
<td>DOUBLE</td>
<td>Displays the collision rate on exclusive locks percentage.</td>
</tr>
<tr>
<td>LAST_EXCLUSIVE_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the last time of blocking exclusive lock calls in microseconds.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_EXCLUSIVE_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of blocking exclusive lock calls in microseconds.</td>
</tr>
<tr>
<td>MIN_EXCLUSIVE_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of blocking exclusive lock calls in microseconds.</td>
</tr>
<tr>
<td>SUM_EXCLUSIVE_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of blocking exclusive lock calls in microseconds.</td>
</tr>
<tr>
<td>AVG_EXCLUSIVE_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of blocking exclusive lock calls in microseconds.</td>
</tr>
<tr>
<td>INTENT_LOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of intent lock calls.</td>
</tr>
<tr>
<td>INTENT_WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of blocking intent lock calls.</td>
</tr>
<tr>
<td>INTENT_CAS_COLLISION_COUNT</td>
<td>BIGINT</td>
<td>Displays the collision count on atomic operation on intent lock.</td>
</tr>
<tr>
<td>INTENT_TIMEOUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of timed out intent lock calls.</td>
</tr>
<tr>
<td>INTENT_COLLISION_RATE</td>
<td>DOUBLE</td>
<td>Displays the collision rate on intent locks percentage.</td>
</tr>
<tr>
<td>LAST_INTENT_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the last time of blocking intent lock calls in microseconds.</td>
</tr>
<tr>
<td>MAX_INTENT_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of blocking intent lock calls in microseconds.</td>
</tr>
<tr>
<td>MIN_INTENT_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of blocking intent lock calls in microseconds.</td>
</tr>
<tr>
<td>SUM_INTENT_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of blocking intent lock calls in microseconds.</td>
</tr>
<tr>
<td>AVG_INTENT_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of blocking intent lock calls in microseconds.</td>
</tr>
<tr>
<td>SHARED_LOCK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of shared lock calls.</td>
</tr>
<tr>
<td>SHARED_WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of blocking shared lock calls.</td>
</tr>
<tr>
<td>SHARED_CAS_COLLISION_COUNT</td>
<td>BIGINT</td>
<td>Displays the collision count on atomic operation on shared lock.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SHARED_TIMEOUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of timed out shared lock calls.</td>
</tr>
<tr>
<td>SHARED_COLLISION_RATE</td>
<td>DOUBLE</td>
<td>Displays the collision rate on shared lock percentage.</td>
</tr>
<tr>
<td>LAST_SHARED_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the last time of blocking shared lock calls in microseconds.</td>
</tr>
<tr>
<td>MAX_SHARED_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of blocking shared lock calls in microseconds.</td>
</tr>
<tr>
<td>MIN_SHARED_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of blocking shared lock calls in microseconds.</td>
</tr>
<tr>
<td>SUM_SHARED_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of blocking shared lock calls in microseconds.</td>
</tr>
<tr>
<td>AVG_SHARED_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of blocking shared lock calls in microseconds.</td>
</tr>
<tr>
<td>COLLISION_RATE</td>
<td>DOUBLE</td>
<td>Displays the global collision rate percentage.</td>
</tr>
<tr>
<td>CREATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of read and write lock creation (for shared statistics only).</td>
</tr>
<tr>
<td>DESTROY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of read/write lock destructions (for shared statistics only).</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VARCHAR(32)</td>
<td>Displays the component.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view contains information about single, or groups, of reader and writer lock objects. It does not contain information about all reader and writer lock. Information like LOCK_COUNT, WAIT_COUNT, and WAIT_TIMES can be used to analyze performance bottlenecks.

This view has a resettable counterpart; you can see the values since the last reset in the M_READWRITELOCKS_RESET system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_READWRITELOCKS_RESET;`. 
6.2.220 M_READWRITELOCKS_RESET System View

Provides read/write lock statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_READWRITELOCKS. Refer to M_READWRITELOCKS for information about the structure and use of this view.

In addition to the members mentioned in M_READWRITELOCKS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

6.2.221 M_RECORD_LOCKS System View

Provides the record lock status.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>LOCK_OWNER_TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction object ID owning the lock.</td>
</tr>
<tr>
<td>LOCK_OWNER_UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the write transaction ID owning the lock.</td>
</tr>
<tr>
<td>ACQUIRED_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the lock acquisition time.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>RECORD_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the record ID.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>LOCK_MODE</td>
<td>VARCHAR(32)</td>
<td>Displays the lock mode (exclusive lock).</td>
</tr>
</tbody>
</table>

**Related Information**

LOCK TABLE Statement (Transaction Management) [page 1056]
Thread Monitoring
Lock Parameters Against Editing for a Tenant Database

### 6.2.222 M_RECOVERY_PROGRESS System View

Shows the progress of the most recent recoveries of tenant databases. M_RECOVERY_PROGRESS is in the SYS_DATABASES schema of a system database.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE_NAME</td>
<td>VARCHAR(256)</td>
<td>Name of the tenant database.</td>
</tr>
<tr>
<td>BACKUP_ID</td>
<td>BIGINT</td>
<td>The unique ID of a recovery.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>The host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>SMALLINT</td>
<td>The internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>The name of the service.</td>
</tr>
<tr>
<td>SYS_START_TIME</td>
<td>TIMESTAMP</td>
<td>The time the recovery was started (local server time)</td>
</tr>
<tr>
<td>UTC_START_TIME</td>
<td>TIMESTAMP</td>
<td>The time the recovery was started (UTC)</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SYS_END_TIME</td>
<td>TIMESTAMP</td>
<td>The time the recovery was completed (local server time)</td>
</tr>
<tr>
<td>UTC_END_TIME</td>
<td>TIMESTAMP</td>
<td>The time the recovery was completed (UTC)</td>
</tr>
<tr>
<td>STATE_NAME</td>
<td>VARCHAR(64)</td>
<td>The current state of the recovery</td>
</tr>
<tr>
<td>PHASE_NAME</td>
<td>VARCHAR(64)</td>
<td>The name of the current phase</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A phase can be the databackup recovery, log backup recovery, restart, or and add or remove a service (if applicable).</td>
</tr>
<tr>
<td>MAX_PHASE</td>
<td>TINYINT</td>
<td>The number of phases to pass</td>
</tr>
<tr>
<td>CURRENT_PHASE</td>
<td>TINYINT</td>
<td>The number of the current phase</td>
</tr>
<tr>
<td>SYS_REACHED_TIME</td>
<td>TIMESTAMP</td>
<td>Point in time reached (local server time)</td>
</tr>
<tr>
<td>UTC_REACHED_TIME</td>
<td>TIMESTAMP</td>
<td>Point in time reached (UTC)</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>VARCHAR(512)</td>
<td>Message that provides a possible reason as to why a recovery failed.</td>
</tr>
<tr>
<td>MAX_PROGRESS</td>
<td>BIGINT</td>
<td>Number of bytes still to be recovered.</td>
</tr>
<tr>
<td>CURRENT_PROGRESS</td>
<td>BIGINT</td>
<td>Number of bytes already recovered.</td>
</tr>
</tbody>
</table>

**Related Information**

M_BACKUP_CATALOG System View [page 1748]
6.2.223 M_REMOTE_CONNECTIONS System View

Provides detailed information on remote connections between databases and remote sources. The information includes connection status, adapter name, and adapter properties.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the adapter name.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source user name.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the connected time.</td>
</tr>
<tr>
<td>CONNECTION_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the connection status: CONNECTED/DISCONNECTED.</td>
</tr>
<tr>
<td>DETAILS</td>
<td>VARCHAR(512)</td>
<td>Displays the adapter properties.</td>
</tr>
<tr>
<td>STATEMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of executed statements.</td>
</tr>
</tbody>
</table>

**Related Information**

Pool of Remote Connections
Configure the Pool of Remote Connections Parameter
M_CONNECTIONS System View [page 1785]
CONNECT Statement (Session Management) [page 721]
6.2.224  M_REMOTE_SOURCES System View

Provides remote source information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>LAST_REFRESH_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last successful completion timestamp of the refresh operation.</td>
</tr>
<tr>
<td>REFRESH_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of when the refresh operation was executed.</td>
</tr>
<tr>
<td>REFRESH_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the refresh operation status: STARTED, COMPLETED, RUNNING (GET OBJECTS), RUNNING (GET OBJECT DETAILS), FAILED, CANCELLED, or CLEARED.</td>
</tr>
<tr>
<td>REFRESH_ERROR_MESSAGE</td>
<td>NVARCHAR(2000)</td>
<td>Displays the exception message that occurred during the refresh operation.</td>
</tr>
</tbody>
</table>

**Related Information**

REMOTE_SOURCES System View [page 1626]
M_REMOTE_SOURCE_LATENCY_HISTORY System View [page 2068]
M_REMOTE_SOURCE_LATENCY_STATUS System View [page 2068]
M_REMOTE_SOURCE_STATISTICS System View [page 2069]
REMOTE_SOURCEOBJECTS System View [page 1627]
REMOTE_SOURCE_OBJECT_COLUMNS System View [page 1629]
REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINTS System View [page 1630]
REMOTE_SOURCE_OBJECT_COLUMN_DESCRIPTIONS System View [page 1631]
REMOTE_SOURCE_OBJECT_DESCRIPTIONS System View [page 1632]
Creating Remote Sources
Customizing the Behavior of a Remote Source

SAP HANA SQL Reference Guide for SAP HANA Platform
System Views Reference
6.2.225 M_REMOTE_SOURCE_LATENCY_HISTORY System View

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>LATENCY_TICKET_NAME</td>
<td>NVARCHAR(32)</td>
<td>Displays the latency ticket name.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>NVARCHAR(32)</td>
<td>Displays the component.</td>
</tr>
<tr>
<td>SUB_COMPONENT</td>
<td>NVARCHAR(32)</td>
<td>Displays the subcomponent.</td>
</tr>
<tr>
<td>STATISTIC_NAME</td>
<td>NVARCHAR(128)</td>
<td>Displays the statistic name.</td>
</tr>
<tr>
<td>STATISTIC_VALUE</td>
<td>NVARCHAR(64)</td>
<td>Displays the statistic value.</td>
</tr>
</tbody>
</table>

6.2.226 M_REMOTE_SOURCE_LATENCY_STATUS System View

Provides remote source latency status information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>LATENCY_TICKET_NAME</td>
<td>NVARCHAR(32)</td>
<td>Displays the latency ticket name.</td>
</tr>
<tr>
<td>INTERVAL</td>
<td>INTEGER</td>
<td>Displays the latency interval.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(8)</td>
<td>Displays the latency ticket status.</td>
</tr>
</tbody>
</table>
6.2.227 M_REMOTE_SOURCE_STATISTICS System View

Returns the remote source operational statistics for monitoring data provisioning components.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription name.</td>
</tr>
<tr>
<td>COLLECT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the collection time.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>NVARCHAR(32)</td>
<td>Displays the component.</td>
</tr>
<tr>
<td>SUB_COMPONENT</td>
<td>NVARCHAR(32)</td>
<td>Displays the subcomponent.</td>
</tr>
<tr>
<td>STATISTIC_NAME</td>
<td>NVARCHAR(128)</td>
<td>Displays the name of the statistic.</td>
</tr>
<tr>
<td>STATISTICS_VALUE</td>
<td>NVARCHAR(512)</td>
<td>Displays the statistic value.</td>
</tr>
</tbody>
</table>

6.2.228 M_REMOTE_STATEMENTS System View

Displays detailed information about executed remote queries. This information includes the query status and the number of fetched rows.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction ID.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>REMOTE_CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the remote connection.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the statement start time.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the statement end time.</td>
</tr>
<tr>
<td>FETCHED_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of fetched records.</td>
</tr>
<tr>
<td>FETCHED_SIZE</td>
<td>BIGINT</td>
<td>Displays the byte size of fetched records.</td>
</tr>
<tr>
<td>REMOTE_DURATION</td>
<td>BIGINT</td>
<td>Displays the duration, in milliseconds, of the total remote request (open/ fetch/close).</td>
</tr>
<tr>
<td>REMOTE_STATEMENT_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the statement status: EXECUTING/CLOSED.</td>
</tr>
<tr>
<td>REMOTE_STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>REMOTE_STATEMENT_DETAILS</td>
<td>NCLOB</td>
<td>Displays the statement details.</td>
</tr>
</tbody>
</table>

### 6.2.229 M_REMOTE_SUBSCRIPTIONS System View

Provides the status and run-time information of a remote subscription.

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote subscription schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote subscription name.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>Column</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STATE</td>
<td>VARCHAR(256)</td>
<td>Displays the event state:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CREATED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MAT_START_BEG_MARKER - Materialization started. Waiting for BEGIN MARKER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MAT_START_END_MARKER - Materialization started. Waiting for end marker</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MAT_COMP_BEG_MARKER - Materialization completed. Waiting for begin marker</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MAT_COMP_END_MARKER - Materialization completed. Waiting for end marker</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AUTO_CORRECT_CHANGE_DATA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• APPLY_CHANGE_DATA</td>
</tr>
<tr>
<td>OPTIMIZED_QUERY_STRING</td>
<td>NCLOB</td>
<td>Displays the optimized query string. If there are multiple subscriptions interested in the same query result, with the same internal distribution ID, each subscription can use the same result.</td>
</tr>
<tr>
<td>INTERNAL_DISTRIBUTION_ID</td>
<td>BIGINT</td>
<td>Displays the generated integer that identifies if multiple target tables are interested in the changes from the same source SQL or virtual table.</td>
</tr>
<tr>
<td>OPTIMIZED_QUERY_RESULTSET_TYPE</td>
<td>TINYINT</td>
<td>Displays the query result set type:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 - REGULAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - CLUSTER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - POOL</td>
</tr>
<tr>
<td>REMOTE_SUBSCRIPTION</td>
<td>NVARCHAR(256)</td>
<td>Displays an optional subscription name registered by the adapter in the remote source system.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>BEGIN_MARKER</td>
<td>VARCHAR(64)</td>
<td>Displays the generated begin marker when the QUEUE command is called. The begin marker must use the format: B&lt;remote_source_oid&gt;<em>&lt;remote_subscription_oid&gt;</em>&lt;YYYYMMDDHH24MMSSFF7&gt;.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>END_MARKER</td>
<td>VARCHAR(64)</td>
<td>Displays the generated end marker when the DISTRIBUTE command is called. The end marker must use the format: E&lt;remote_source_oid&gt;_&lt;remote_subscription_oid&gt;_YYYYMMDDHH24MMSSFF7.</td>
</tr>
<tr>
<td>BEGIN_MARKER_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the QUEUE request is received.</td>
</tr>
<tr>
<td>END_MARKER_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the DISTRIBUTE command is called.</td>
</tr>
<tr>
<td>LAST_PROCESSED_TRANSACTION_ID</td>
<td>VARBINARY(128)</td>
<td>Displays the transaction ID of the last processed transaction.</td>
</tr>
<tr>
<td>LAST_PROCESSED_TRANSACTION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time when the last transaction was applied.</td>
</tr>
<tr>
<td>LAST_PROCESSED_BEGIN_SEQUENCE_ID</td>
<td>VARBINARY(68)</td>
<td>Displays the begin record sequence ID of the last processed transaction.</td>
</tr>
<tr>
<td>LAST_PROCESSED_COMMIT_SEQUENCE_ID</td>
<td>VARBINARY(68)</td>
<td>Displays the commit record sequence ID of the last processed transaction.</td>
</tr>
<tr>
<td>LAST_RECEIVED_SEQUENCE_ID</td>
<td>VARBINARY(68)</td>
<td>Displays the last received sequence ID.</td>
</tr>
<tr>
<td>LAST_RECEIVED-custom_id</td>
<td>NVARCHAR(64)</td>
<td>Displays the last received custom ID. Custom IDs may be used by adapters with every changed-data row of a transaction.</td>
</tr>
<tr>
<td>LAST_PROCESSEDustom_id</td>
<td>NVARCHAR(64)</td>
<td>Displays the last processed custom ID. Custom IDs may be used by adapters with every changed-data row of a transaction.</td>
</tr>
</tbody>
</table>

### 6.2.230 M_REMOTE_SUBSCRIPTION_COMMIT_SEQUENCE_CONTAINERS System View

Lists all real time data elements such as markers and commit / rollback rows in the remote subscription commit sequence container.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SOURCE_OID</td>
<td>BIGINT</td>
<td>Displays the remote source OID.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_OID</td>
<td>BIGINT</td>
<td>Displays the subscription OID.</td>
</tr>
<tr>
<td>SUBSCRIPTION_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription name.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID of the virtual file.</td>
</tr>
<tr>
<td>ELEMENT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the element type. Values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - BEGIN MARKER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - END MARKER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - COMMITTED TRANSACTION</td>
</tr>
<tr>
<td>MARKER</td>
<td>VARCHAR(64)</td>
<td>Displays the marker string.</td>
</tr>
<tr>
<td>SEQUENCE_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the sequence ID of the row.</td>
</tr>
<tr>
<td>ROW_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the internal CDC row type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – commit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – rollback</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 – begin-marker</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 – end-marker</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the remote source transaction ID of the row.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
</tbody>
</table>
## 6.2.231 M_REMOTE_SUBSCRIPTION_COMMIT_SEQUENCE_GROUP_CONTAINERS System View

Lists all CommitSequece virtual files for a remote source, and the number of entries stored in each virtual file.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SOURCE_OID</td>
<td>BIGINT</td>
<td>Displays the remote source OID.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_OID</td>
<td>BIGINT</td>
<td>Displays the subscription OID</td>
</tr>
<tr>
<td>SUBSCRIPTION_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription name.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the container of the virtual file.</td>
</tr>
<tr>
<td>COMMIT_SEQUENCE_CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID of the CommitSequence virtual file.</td>
</tr>
<tr>
<td>MIN_SEQUENCE_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the minimum sequence ID of the CommitSequence virtual file.</td>
</tr>
<tr>
<td>MAX_SEQUENCE_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the maximum sequence ID of the CommitSequence virtual file.</td>
</tr>
<tr>
<td>ENTRY_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of CommitSequence entries in the virtual file.</td>
</tr>
<tr>
<td>IS_DELETED</td>
<td>VARCHAR(5)</td>
<td>Indicates if the container identified by COMMIT_SEQUENCE_CONTAINER_ID is deleted. Values are TRUE or FALSE.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
</tbody>
</table>
6.2.232  M_REMOTE_SUBSCRIPTION_COMPONENTS System View

Provides remote subscription component information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription name.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VARCHAR(10)</td>
<td>Displays the component: DPSERVER, ADAPTER, RECEIVER, or APPLIER.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(10)</td>
<td>Displays the component status.</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>NVARCHAR(2000)</td>
<td>Displays an error or warning message.</td>
</tr>
</tbody>
</table>

6.2.233  M_REMOTE_SUBSCRIPTION_DISTRIBUTOR_QUEUE_DATA_CONTAINERS System View

Lists all real time data elements between begin-marker and end-marker for the remote subscription.

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SOURCE_OID</td>
<td>BIGINT</td>
<td>Displays the remote source OID.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_OID</td>
<td>BIGINT</td>
<td>Displays the subscription OID.</td>
</tr>
<tr>
<td>SUBSCRIPTION_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription name.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID of this virtual file.</td>
</tr>
<tr>
<td>Column</td>
<td>Data type</td>
<td>Purpose</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ELEMENT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the element type. Values are: 1 - BEGIN MARKER, 2 - END MARKER, 3 - COMMITTED TRANSACTION</td>
</tr>
<tr>
<td>MARKER</td>
<td>VARCHAR(64)</td>
<td>Displays the marker string.</td>
</tr>
<tr>
<td>SEQUENCE_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the sequence ID of the row.</td>
</tr>
<tr>
<td>ROW_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the internal CDC row type. Values are: 2 – COMMIT, 3 – ROLLBACK, 8 – BEGIN MARKER, 9 – END MARKER</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the remote source transaction ID of the row.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
</tbody>
</table>

### M_REMOTE_SUBSCRIPTION_LOB_CONTAINERS System View

Lists all lob container IDs for each remote subscription transaction.

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SOURCE_OID</td>
<td>BIGINT</td>
<td>Displays the remote source OID.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_OID</td>
<td>BIGINT</td>
<td>Displays the subscription OID.</td>
</tr>
<tr>
<td>SUBSCRIPTION_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription name.</td>
</tr>
<tr>
<td>Column</td>
<td>Data type</td>
<td>Purpose</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID of this virtual file.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the remote source transaction ID of the lob.</td>
</tr>
<tr>
<td>LOB_CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID of the lob virtual file.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
</tbody>
</table>

**6.2.235 M_REMOTE_SUBSCRIPTION_ROLLBACK_SAVEPOINT_CONTAINERS System View**

Lists all rollback save points for each remote subscription transaction.

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SOURCE_OID</td>
<td>BIGINT</td>
<td>Displays the remote source OID.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_OID</td>
<td>BIGINT</td>
<td>Displays the subscription OID.</td>
</tr>
<tr>
<td>SUBSCRIPTION_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription name.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID of this virtual file.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the remote source transaction ID of the row.</td>
</tr>
<tr>
<td>FROM_SEQUENCE_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the sequence ID of the row to roll back from.</td>
</tr>
<tr>
<td>TO_SEQUENCE_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the sequence ID of the row to roll back to.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>Column</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
</tbody>
</table>

### 6.2.236 M_REMOTE_SUBSCRIPTION_STATISTICS System View

Provides remote subscription statistic information.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription name.</td>
</tr>
<tr>
<td>RECEIVED_MESSAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the received message count.</td>
</tr>
<tr>
<td>RECEIVED_MESSAGE_SIZE</td>
<td>BIGINT</td>
<td>Displays the received message size in bytes.</td>
</tr>
<tr>
<td>APPLIED_MESSAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the received message count.</td>
</tr>
<tr>
<td>APPLIED_MESSAGE_SIZE</td>
<td>BIGINT</td>
<td>Displays the applied message size in bytes.</td>
</tr>
<tr>
<td>REJECTED_MESSAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the rejected message count.</td>
</tr>
<tr>
<td>LAST_MESSAGE_RECEIVED</td>
<td>TIMESTAMP</td>
<td>Displays the last received message.</td>
</tr>
<tr>
<td>LAST_MESSAGE_APPLIED</td>
<td>TIMESTAMP</td>
<td>Displays the last applied message.</td>
</tr>
<tr>
<td>RECEIVER_LATENCY</td>
<td>BIGINT</td>
<td>Displays the receiver latency.</td>
</tr>
<tr>
<td>APPLIER_LATENCY</td>
<td>BIGINT</td>
<td>Displays the applier latency.</td>
</tr>
</tbody>
</table>
### 6.2.237 M_REMOTE_SUBSCRIPTION_TRANSACTION_CONTA INERS System View

Lists all real time data rowsets in the remote subscription transaction container.

<table>
<thead>
<tr>
<th>Column</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_SOURCE_OID</td>
<td>BIGINT</td>
<td>Displays the remote source OID.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_OID</td>
<td>BIGINT</td>
<td>Displays the subscription OID.</td>
</tr>
<tr>
<td>SUBSCRIPTION_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription schema name.</td>
</tr>
<tr>
<td>SUBSCRIPTION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the subscription name.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID of the virtual file.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the remote source transaction ID of the row.</td>
</tr>
<tr>
<td>SEQUENCE_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the sequence ID of the row.</td>
</tr>
<tr>
<td>Column</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ROW_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the internal CDC row type. Values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – BEGIN TRANSACTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – COMMIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 – ROLLBACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 – INSERT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 – DELETE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 – BEFORE-IMAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 – AFTER-IMAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 – BEGIN-MARKER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 – END-MARKER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 – ROLLBACK-TO-SAVEPOINT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 – REPLACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 – SCHEMA-CHANGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 – TRUNCATE-TABLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14 – UPSERT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15 – TRUNCATE-REPLACE-TARGET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16 – BEGIN-REPLACE-SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17 – END-REPLACE-SET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18 – EXTERMINATE-ROW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 – LATENCY-STATISTICS</td>
</tr>
<tr>
<td>DIGEST</td>
<td>VARCHAR(1000)</td>
<td>Displays the digest of the row content.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
</tbody>
</table>
## 6.2.238 M_REMOTE_TABLE_REPLICAS System View

Provides remote table replication information.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistent volume ID of the server.</td>
</tr>
<tr>
<td>IS_SOURCE_SYSTEM</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the system has a source table: TRUE/FALSE.</td>
</tr>
<tr>
<td>REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name on the replication system.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the replication table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of the replication table.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the Table OID of the replication table.</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of replication table: ROW/COLUMN.</td>
</tr>
<tr>
<td>REPLICA_TYPE</td>
<td>VARCHAR(12)</td>
<td>Displays the replication type: SYNCHRONOUS/ASYNCHRONOUS.</td>
</tr>
<tr>
<td>SOURCE_REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name on the source system.</td>
</tr>
<tr>
<td>SOURCE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the source table.</td>
</tr>
<tr>
<td>SOURCE_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of the source table.</td>
</tr>
<tr>
<td>SOURCE_TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the table OID of a source table.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SOURCE_TABLE_TYPE</td>
<td>VARCHAR(6)</td>
<td>Displays the source table type: ROW/COLUMN.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is in progress.</td>
</tr>
<tr>
<td>REPLICATION_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the replication status: ENABLING, ENABLED, or DISABLED.</td>
</tr>
<tr>
<td>LAST_ENABLE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the table was last enabled.</td>
</tr>
<tr>
<td>LAST_DISABLE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the table was last disabled.</td>
</tr>
<tr>
<td>LAST_ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the last error code number.</td>
</tr>
<tr>
<td>LAST_ERROR_MESSAGE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the last error message.</td>
</tr>
<tr>
<td>LAST_ERROR_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the last error occurred.</td>
</tr>
<tr>
<td>INSERT_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of inserted records.</td>
</tr>
<tr>
<td>UPDATE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of updated records.</td>
</tr>
<tr>
<td>DELETE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of deleted records.</td>
</tr>
<tr>
<td>INSERT_RETRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of retries to insert.</td>
</tr>
<tr>
<td>UPDATE_RETRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of retries to update.</td>
</tr>
<tr>
<td>DELETE_RETRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of retries to delete.</td>
</tr>
</tbody>
</table>
6.2.239 M_REORG_ALGORITHMS System View

Provides information about landscape redistribution algorithms.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALGORITHM_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the algorithm.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR(256)</td>
<td>Displays the description.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>VARCHAR(256)</td>
<td>Displays any optional parameters.</td>
</tr>
</tbody>
</table>

6.2.240 M_REPLICATION_LOG System View

Provides replication log monitoring information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID.</td>
</tr>
<tr>
<td>IS_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays TRUE when the replication logs are being collected and FALSE when collecting replication logs is disabled.</td>
</tr>
<tr>
<td>LAST_DISABLE_REASON</td>
<td>VARCHAR(32)</td>
<td>Displays the latest disable reason: REPLICATION_LOG_SPACE_FULL/USER_COMMAND.</td>
</tr>
<tr>
<td>USED_REPLICATION_LOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the used replication log size in the volume in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_REPLICATION_LOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the limitation of the replication log size in the volume in bytes.</td>
</tr>
</tbody>
</table>

### 6.2.241 M_REPO_TRANSPORT_FILES System View

Provides information about all repository transport files.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the file name.</td>
</tr>
<tr>
<td>FILE_SIZE</td>
<td>BIGINT</td>
<td>Displays the file size in bytes.</td>
</tr>
<tr>
<td>FILE_MTIME</td>
<td>TIMESTAMP</td>
<td>Displays the file date.</td>
</tr>
</tbody>
</table>

### 6.2.242 M_RESULT_CACHE System View

Provides result cache information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CACHE_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID for each result cache entry.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>DETAILS</td>
<td>NCLOB</td>
<td>Displays the details.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of memory occupied by the result cache entry in bytes.</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of records in the cache entry.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp the result cache entry is generated and stored.</td>
</tr>
<tr>
<td>LAST_REFRESH_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last refresh time of the result cache entry.</td>
</tr>
<tr>
<td>LAST_REFRESH_DURATION</td>
<td>BIGINT</td>
<td>Displays the time duration of the cache entry preparation in milliseconds.</td>
</tr>
<tr>
<td>REFRESH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of refreshments on the cache entry.</td>
</tr>
<tr>
<td>LAST_ACCESS_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time the result cache entry was accessed by a user.</td>
</tr>
<tr>
<td>ACCESS_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of accesses on the cache entry.</td>
</tr>
<tr>
<td>RUNTIME_MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of cache miss counts. This value is decided at runtime with parameter values.</td>
</tr>
<tr>
<td>EVICT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of eviction counts due to budget limitations.</td>
</tr>
<tr>
<td>IS_EVICTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the cache entry is evicted: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
6.2.243  M_RESULT_CACHE_EXCLUSIONS System View

Provides information about result cache exclusions.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type.</td>
</tr>
<tr>
<td>REASON</td>
<td>NVARCHAR(1024)</td>
<td>Displays the reason why the object cannot use result cache.</td>
</tr>
<tr>
<td>EXCLUDE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when disabling the result cache was decided.</td>
</tr>
</tbody>
</table>

Related Information

M_RESULT_CACHE System View [page 2084]

6.2.244  M_RESULT_CACHE_RESET System View

Provides information about result cache statistics.

This view contains values accumulated since the last reset of the main view M_RESULT_CACHE. Refer to M_RESULT_CACHE for information about the structure and use of this view.

In addition to the members mentioned in M_RESULT_CACHE, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_RESULT_CACHE System View [page 2084]
6.2.245 M_RS_INDEXES System View

Provides the statistics for B-tree and CPB-tree indexes.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the index name.</td>
</tr>
<tr>
<td>KEY_TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the key type (data or composite).</td>
</tr>
<tr>
<td>INDEX_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the index status: VALID, INVALID, or UNUSABLE.</td>
</tr>
<tr>
<td>TREE_HEIGHT</td>
<td>BIGINT</td>
<td>Displays the B-tree level.</td>
</tr>
<tr>
<td>LEAF_NODE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of leaf nodes.</td>
</tr>
<tr>
<td>NONLEAF_NODE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of intermediate nodes.</td>
</tr>
<tr>
<td>NODE_SIZE</td>
<td>BIGINT</td>
<td>Displays the node size in bytes.</td>
</tr>
<tr>
<td>FANOUT</td>
<td>DOUBLE</td>
<td>Displays the index fan-out.</td>
</tr>
<tr>
<td>BULKLOAD_FACTOR</td>
<td>DOUBLE</td>
<td>Displays the fill factor for creating or recovering indexes.</td>
</tr>
<tr>
<td>INDEX_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory index size (number of nodes * node size) in bytes.</td>
</tr>
<tr>
<td>ENTRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of indexed records.</td>
</tr>
<tr>
<td>FIXED_LEAF_NODE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of fixed leaf nodes. FIXED means that the key lengths of all entries in the node have the same value.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FIXED_NONLEAF_NODE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of fixed non-leaf nodes.</td>
</tr>
<tr>
<td>AVG_LEAF_OFFSET_SIZE</td>
<td>DOUBLE</td>
<td>Displays the average offset size of leaf nodes.</td>
</tr>
<tr>
<td>AVG_NONLEAF_OFFSET_SIZE</td>
<td>DOUBLE</td>
<td>Displays the average offset size of non-leaf nodes.</td>
</tr>
<tr>
<td>AVG_LEAF_POINTER_SIZE</td>
<td>DOUBLE</td>
<td>Displays the average pointer size of leaf nodes.</td>
</tr>
<tr>
<td>AVG_NONLEAF_POINTER_SIZE</td>
<td>DOUBLE</td>
<td>Displays the average pointer size of non-leaf nodes.</td>
</tr>
<tr>
<td>LEAF_PARTIAL_KEY_SIZE</td>
<td>BIGINT</td>
<td>Displays the partial key length of the leaf node.</td>
</tr>
<tr>
<td>NONLEAF_PARTIAL_KEY_SIZE</td>
<td>BIGINT</td>
<td>Displays the partial key length of the non-leaf node.</td>
</tr>
<tr>
<td>INDEX_UTILIZATION</td>
<td>DOUBLE</td>
<td>Displays the index utilization (num_entries / max_entries).</td>
</tr>
<tr>
<td>IS_UNIQUE</td>
<td>VARCHAR(5)</td>
<td>Displays the whether the index is unique: TRUE or FALSE.</td>
</tr>
<tr>
<td>SEARCH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of search operations.</td>
</tr>
<tr>
<td>INSERT_COUNT</td>
<td>BIGINT</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>REMOVE_COUNT</td>
<td>BIGINT</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>FULL_KEY_REFERENCE_COUNT</td>
<td>BIGINT</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>DISTINCT_KEY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of distinct keys.</td>
</tr>
<tr>
<td>KEY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of keys.</td>
</tr>
<tr>
<td>ELIMINATED_DUPLICATE_LEAF_NODE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of leaf nodes whose duplicate keys are eliminated.</td>
</tr>
<tr>
<td>UNUSED_LEAF_SLOTS_PER_NODE</td>
<td>DOUBLE</td>
<td>Displays the number of unused slots per leaf node.</td>
</tr>
</tbody>
</table>
6.2.246  M_RS_MEMORY System View

Provides RS memory statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>VARCHAR(128)</td>
<td>Displays the module name.</td>
</tr>
<tr>
<td>ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated memory size of the module in bytes.</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the used memory size of the module in bytes.</td>
</tr>
<tr>
<td>FREE_SIZE</td>
<td>BIGINT</td>
<td>Displays the free memory size of the module in bytes.</td>
</tr>
</tbody>
</table>

6.2.247  M_RS_TABLES System View

Provides information about row tables, including detailed table sizes and record count.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of records in this table.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ALLOCATED_FIXED_PART_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated memory size in bytes for the fixed-size part.</td>
</tr>
<tr>
<td>FIXED_PAGE_HEADER_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated and used memory size in bytes for the page headers of the fixed-size part.</td>
</tr>
<tr>
<td>FIXED_PAGE_FRAGMENT_SIZE</td>
<td>BIGINT</td>
<td>Displays the fragmented memory size in bytes of the fixed-size part.</td>
</tr>
<tr>
<td>USED_FIXED_PART_SIZE</td>
<td>BIGINT</td>
<td>Displays the used memory size in bytes for the fixed-size part.</td>
</tr>
<tr>
<td>FIXED_PART_FRAGMENT_SIZE</td>
<td>BIGINT</td>
<td>Displays the fragmented memory size in bytes of the used fixed-size part.</td>
</tr>
<tr>
<td>FIXED_PART_FREE_SIZE</td>
<td>BIGINT</td>
<td>Displays the free memory size in bytes in the pages of the fixed-size part.</td>
</tr>
<tr>
<td>ALLOCATED_VARIABLE_PART_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated memory size in bytes for the variable-size part.</td>
</tr>
<tr>
<td>USED_VARIABLE_PART_SIZE</td>
<td>BIGINT</td>
<td>Displays the used memory size in bytes for the variable-size part.</td>
</tr>
<tr>
<td>VARIABLE_PART_FRAGMENT_SIZE</td>
<td>BIGINT</td>
<td>Displays the fragmented memory size in bytes of the used variable-size part.</td>
</tr>
<tr>
<td>LOAD_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the loading state of the table: LOADED, NOW_LOADING, PREPARING_UNLOAD, NOW_UNLOADING, UNLOADED, or NOT_SUPPORTED</td>
</tr>
<tr>
<td>IS_REPLICA</td>
<td>VARCHAR(5)</td>
<td>Displays the flag to indicate a replica.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the create time of the table.</td>
</tr>
<tr>
<td>CONTAINER_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of containers for the table.</td>
</tr>
<tr>
<td>SCAN_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of table scans.</td>
</tr>
</tbody>
</table>
6.2.248 M_RS_TABLE_VERSION_STATISTICS System View

Provides information on row table versions: detailed version counts and used sizes.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema to which the container belongs.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of the container.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>INTEGER</td>
<td>Indicates that some tables may have several containers to store its data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The container ID is a unique identifier for each container.</td>
</tr>
<tr>
<td>IS_SYSTEM_TABLE</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the table is a system table: TRUE/FALSE.</td>
</tr>
<tr>
<td>VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of all versions in the container.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This includes all versions regardless of their version type and status.</td>
</tr>
<tr>
<td>INSERT_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of insert-versions in the container.</td>
</tr>
<tr>
<td>UPDATE_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of update-versions in the container.</td>
</tr>
<tr>
<td>DELETE_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of delete-versions in the container.</td>
</tr>
<tr>
<td>COMMITTED_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of all committed versions in the container (includes all types).</td>
</tr>
<tr>
<td>COMMITTED_INSERT_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of committed insert-versions in the container.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COMMITTED_UPDATE_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of committed update-versions in the container.</td>
</tr>
<tr>
<td>COMMITTED_DELETE_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of committed delete-versions in the container.</td>
</tr>
<tr>
<td>UNCOMMITTED_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of all uncommitted versions in the container (includes all types).</td>
</tr>
<tr>
<td>UNCOMMITTED_INSERT_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of uncommitted insert-versions in the container.</td>
</tr>
<tr>
<td>UNCOMMITTED_UPDATE_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of uncommitted update-versions in the container.</td>
</tr>
<tr>
<td>UNCOMMITTED_DELETE_VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of uncommitted delete-versions in the container.</td>
</tr>
<tr>
<td>VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of the versions in the container in bytes. This includes all versions regardless of their version type and status.</td>
</tr>
<tr>
<td>INSERT_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of the insert-versions in the container in bytes.</td>
</tr>
<tr>
<td>UPDATE_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of update-versions in the container in bytes.</td>
</tr>
<tr>
<td>DELETE_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of delete-versions in the container in bytes.</td>
</tr>
<tr>
<td>COMMITTED_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of all of the committed versions in the container (includes all types) in bytes.</td>
</tr>
<tr>
<td>COMMITTED_INSERT_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of committed insert-versions in the container in bytes.</td>
</tr>
<tr>
<td>COMMITTED_UPDATE_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of committed update-versions in the container in bytes.</td>
</tr>
<tr>
<td>COMMITTED_DELETE_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of committed delete-versions in the container in bytes.</td>
</tr>
<tr>
<td>UNCOMMITTED_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of all uncommitted versions in the container (includes all types) in bytes.</td>
</tr>
</tbody>
</table>
### Column name

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNCOMMITTED_INSERT_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of uncommitted insert-versions in the container in bytes.</td>
</tr>
<tr>
<td>UNCOMMITTED_UPDATE_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of uncommitted update-versions in the container in bytes.</td>
</tr>
<tr>
<td>UNCOMMITTED_DELETE_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the used size of uncommitted delete-versions in the container in bytes.</td>
</tr>
<tr>
<td>MIN_COMMIT_ID</td>
<td>BIGINT</td>
<td>Indicates that each committed version has its commit ID. MIN_COMMIT_ID refers to the minimum value of commit ID among all committed versions.</td>
</tr>
</tbody>
</table>

### 6.2.249 M_SAVEPOINTS System View

Displays current and historical savepoint statistics.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the savepoint start time.</td>
</tr>
<tr>
<td>INITIATION</td>
<td>VARCHAR(24)</td>
<td>Displays the reason why the savepoint was executed.</td>
</tr>
<tr>
<td>PURPOSE</td>
<td>VARCHAR(24)</td>
<td>Displays the purpose of the savepoint.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
---|---|---
STATE | VARCHAR(16) | Displays the savepoint state. A savepoint is triggered periodically, but it can be also triggered on demand using the `ALTER SYSTEM SAVEPOINT` statement or by a data backup. The savepoint then runs through the following various states:

- **INITIAL**: Displays the empty statistics record (just allocated); immediately goes to the PREPARE state.
- **PREPARE**: Displays that the savepoint is preparing to run.
- **PAGEFLUSH**: flushes dirty pages asynchronously. PAGES and SIZE counters are updated to reflect the savepoint progress.
- **PRECRITICAL**: runs any pre-critical-phase callbacks and waits for I/O of the rest of the flushed pages.
- **ENTERCRITICAL**: entering critical phase: stops updaters and waits for the critical phase lock.
- **CRITICAL**: critical phase: copies and triggers the write of any pages that are still dirty, records various information in the restart record (log position, transaction manager state, and so on), and increments the savepoint version. PAGES_IN_CRITICAL_PHASE and SIZE_IN_CRITICAL_PHASE counters are updated appropriately.
- **EXITCRITICAL**: exit critical phase: releases critical phase locks and restarts updaters.
- **POSTCRITICAL**: calls any post-critical phase callbacks, finalizes the savepoint record, and cleans up unneeded RTT entries.
### Column name | Data type | Description
--- | --- | ---
FINISHING  • waits for the flush of pages written in the critical phase and writes out the restart record and anchor page.  • DONE - Displays that the savepoint has been completed successfully.  • ABORTED - Displays that the savepoint has been aborted (for example, due to a timeout during the synchronization of the global savepoint).  
VERSION | INTEGER | Displays the savepoint version.  
REQUESTED_FREQUENCY | BIGINT | Displays the frequency of the savepoint.  
TIME_SINCE_PREVIOUS | BIGINT | Displays the time in seconds between the current and previous current savepoints.  
DURATION | BIGINT | Displays the total time in microseconds to create the savepoint.  
PREPARE_FLUSH_RETRY_COUNT | BIGINT | Displays the number of times the pages were flushed in the non-critical phase before entering the critical phase.  
BLOCKING_PHASE_START_TIME | TIMESTAMP | Displays the start time of the last blocking phase.  
BLOCKING_PHASE_DURATION | BIGINT | Displays the phase duration in microseconds.  
CRITICAL_PHASE_START_TIME | TIMESTAMP | Displays the start time of the last critical phase.  
CRITICAL_PHASE_DURATION | BIGINT | Displays the time in microseconds spent in the critical phase, during which updates are blocked.  
CRITICAL_PHASE_WAIT_TIME | BIGINT | Displays the wait time in microseconds for the critical phase.  
TOTAL_SIZE | BIGINT | Displays the number of bytes that have been prompted to be written to the disk.
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLUSHED_PAGES</td>
<td>BIGINT</td>
<td>Displays the number of asynchronously flushed pages.</td>
</tr>
<tr>
<td>FLUSHED_PAGES_IN_CRITICAL_PHASE</td>
<td>BIGINT</td>
<td>Displays the number of pages flushed in the critical phase.</td>
</tr>
<tr>
<td>FLUSHED_ROWSTORE_PAGES</td>
<td>BIGINT</td>
<td>Displays the number of asynchronously flushed row store pages.</td>
</tr>
<tr>
<td>FLUSHED_ROWSTORE_PAGES_IN_CRITICAL_PHASE</td>
<td>BIGINT</td>
<td>Displays the number of row store pages flushed in the critical phase.</td>
</tr>
<tr>
<td>FLUSHED_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of asynchronously flushed pages in bytes.</td>
</tr>
<tr>
<td>FLUSHED_SIZE_IN_CRITICAL_PHASE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of pages flushed in the critical phase.</td>
</tr>
<tr>
<td>FLUSHED_ROWSTORE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the asynchronously flushed row store pages.</td>
</tr>
<tr>
<td>FLUSHED_ROWSTORE_SIZE_IN_CRITICAL_PHASE</td>
<td>BIGINT</td>
<td>Displays the size of the row store pages flushed in the critical phase.</td>
</tr>
<tr>
<td>RTT_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size in bytes of the rollback transaction table at the savepoint.</td>
</tr>
</tbody>
</table>

### Additional Information

This monitor view contains information about the current and some past savepoints. Information is kept on the last 128 savepoints per persistence manager.

**Note**

Only savepoint information since the startup is kept. The information in this monitor view does not persist. The statistics server collects some information about savepoints that persists.

This view is associated with the ALTER SYSTEM SAVEPOINT statement.

The following information can be extracted from the numbers in this view:

- **CRITICAL_PHASE_DURATION** shows the period of time during which the updaters were blocked in a savepoint. Normally, this should be in the millisecond range, except for global savepoints for data backup, which may take longer due to global synchronization across all nodes. If the critical phase duration is too long, there may be a problem (for example, the I/O load is too high).
• **DURATION** shows the total time taken by a savepoint. This should be significantly less than the configured savepoint frequency `REQUESTED_FREQUENCY` (in the range 0-10%, depending on the load). Higher ratios indicate an I/O overload.

• **TIME_SINCE_PREVIOUS** should be close to `REQUESTED_FREQUENCY`. If it is significantly higher, this indicates that the savepoint is encountering a block, such as a very long column merge operation.

• A ratio of `FLUSHED_PAGES*` vs. `FLUSHED_ROWSTORE_PAGES*` or a ratio of `FLUSHED_SIZE*` vs. `FLUSHED_ROWSTORE_SIZE*` shows the respective load of column store vs. row store. The row store is only flushed during the savepoint, and the column store also flushes the data between savepoints to balance the load.

• A high ratio of `FLUSHED_*PAGES_IN_CRITICAL_PHASE` vs. `FLUSHED_*PAGES` or a ratio of `FLUSHED_*SIZE_IN_CRITICAL_PHASE` vs. `FLUSHED_*SIZE` indicates a potential I/O overload. Normally, zero or only a few pages should be written in the critical phase, except for special situations like a global savepoint for data backups (but in this case, the number of pages written in the critical phase should be in the order of the magnitude, 1% or less, of asynchronously flushed pages). High amounts of data written in the critical phase indicates an overload of the I/O subsystem and can lead to increased blocking times of update transactions due to an increased `CRITICAL_PHASE_DURATION`.

• A big `RTT_SIZE` (more than a few entries) indicates a problem in distributed transaction handling. RTT (rollback transaction table) holds rollback entries for distributed transactions that are currently in rollback. Normally, these entries are eliminated quickly after the respective rollback is finished. In cases where a slave node fails, entries for the slave node are held persistently until the slave node restarts. This number goes to zero or close to zero shortly after the restart of a failed slave node (or after the restart of the whole system).

Aggregated values for individual counters can be queried from the `M_SAVEPOINT_STATISTICS` system view.

### Related Information

- **M_SAVEPOINT_STATISTICS System View** [page 2097]
- **ALTER SYSTEM SAVEPOINT Statement (System Management)** [page 567]

### 6.2.250 M_SAVEPOINT_STATISTICS System View

Provides information about executed savepoints.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence Volume ID.</td>
</tr>
<tr>
<td>SAVEPOINTS</td>
<td>BIGINT</td>
<td>Displays the number of executed savepoints.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time of the last savepoint.</td>
</tr>
<tr>
<td>INITIATION</td>
<td>VARCHAR(24)</td>
<td>Displays the reason why the last savepoint was executed.</td>
</tr>
<tr>
<td>PURPOSE</td>
<td>VARCHAR(24)</td>
<td>Displays the reason why the last savepoint was executed.</td>
</tr>
<tr>
<td>STATE</td>
<td>VARCHAR(16)</td>
<td>Displays the last savepoint state.</td>
</tr>
<tr>
<td>VERSION</td>
<td>INTEGER</td>
<td>Displays the last savepoint version.</td>
</tr>
<tr>
<td>REQUESTED_FREQUENCY</td>
<td>BIGINT</td>
<td>Displays the currently active configured savepoint frequency in seconds.</td>
</tr>
<tr>
<td>LAST_FREQUENCY</td>
<td>BIGINT</td>
<td>Displays the actual frequency in seconds between the last two savepoints.</td>
</tr>
<tr>
<td>AVG_FREQUENCY</td>
<td>BIGINT</td>
<td>Displays the actual average frequency in seconds between last two savepoints.</td>
</tr>
<tr>
<td>LAST_DURATION</td>
<td>BIGINT</td>
<td>Displays the total time spent in microseconds creating the last savepoint.</td>
</tr>
<tr>
<td>AVG_DURATION</td>
<td>BIGINT</td>
<td>Displays the average time spent in microseconds creating the last two save-</td>
</tr>
<tr>
<td>LAST_BLOCKING_PHASE_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time of the last blocking phase.</td>
</tr>
<tr>
<td>LAST_BLOCKING_PHASE_DURATION</td>
<td>BIGINT</td>
<td>Displays the duration in seconds of the last blocking phase.</td>
</tr>
<tr>
<td>LAST_CRITICAL_PHASE_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time of the last critical phase.</td>
</tr>
<tr>
<td>LAST_CRITICAL_PHASE_DURATION</td>
<td>BIGINT</td>
<td>Displays the time spent in microseconds in the last critical phase, during</td>
</tr>
<tr>
<td></td>
<td></td>
<td>which updates are blocked.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AVG_CRITICAL_PHASE_DURATION</td>
<td>BIGINT</td>
<td>Displays the average time spent in microseconds for the last two critical phases, during which updates are blocked.</td>
</tr>
<tr>
<td>LAST_TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the total number of bytes written for the last savepoint.</td>
</tr>
<tr>
<td>AVG_TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the average number of bytes written for a savepoint.</td>
</tr>
<tr>
<td>LAST_FLUSHED_PAGES</td>
<td>BIGINT</td>
<td>Displays the last number of asynchronously flushed pages.</td>
</tr>
<tr>
<td>AVG_FLUSHED_PAGES</td>
<td>DOUBLE</td>
<td>Displays the average number of asynchronously flushed pages.</td>
</tr>
<tr>
<td>LAST_FLUSHED_PAGES_IN_CRITICAL_PHASE</td>
<td>BIGINT</td>
<td>Displays the last number of pages flushed in the critical phase.</td>
</tr>
<tr>
<td>AVG_FLUSHED_PAGES_IN_CRITICAL_PHASE</td>
<td>DOUBLE</td>
<td>Displays the average number of pages flushed in the critical phase.</td>
</tr>
<tr>
<td>LAST_FLUSHED_ROWSTORE_PAGES</td>
<td>BIGINT</td>
<td>Displays the last number of asynchronously flushed row store pages.</td>
</tr>
<tr>
<td>AVG_FLUSHED_ROWSTORE_PAGES</td>
<td>DOUBLE</td>
<td>Displays the average number of asynchronously flushed row store pages.</td>
</tr>
<tr>
<td>LAST_FLUSHED_ROWSTORE_PAGES_IN_CRITICAL_PHASE</td>
<td>BIGINT</td>
<td>Displays the last number of row store pages flushed in the critical phase.</td>
</tr>
<tr>
<td>AVG_FLUSHED_ROWSTORE_PAGES_IN_CRITICAL_PHASE</td>
<td>DOUBLE</td>
<td>Displays the average number of row store pages flushed in the critical phase.</td>
</tr>
<tr>
<td>LAST_FLUSHED_SIZE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the last asynchronously flushed pages.</td>
</tr>
<tr>
<td>AVG_FLUSHED_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size in bytes of asynchronously flushed pages.</td>
</tr>
<tr>
<td>LAST_FLUSHED_SIZE_IN_CRITICAL_PHASE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the last pages flushed in the critical phase.</td>
</tr>
<tr>
<td>AVG_FLUSHED_SIZE_IN_CRITICAL_PHASE</td>
<td>BIGINT</td>
<td>Displays the average size of pages flushed in the critical phase.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LAST_FLUSHED_ROWSTORE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the last asynchronously flushed row store pages.</td>
</tr>
<tr>
<td>AVG_FLUSHED_ROWSTORE_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size in bytes of the asynchronously flushed row store pages.</td>
</tr>
<tr>
<td>LAST_FLUSHED_ROWSTORE_SIZE_IN_CRITICAL_PHASE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the last row store pages flushed in the critical phase.</td>
</tr>
<tr>
<td>AVG_FLUSHED_ROWSTORE_SIZE_IN_CRITICAL_PHASE</td>
<td>BIGINT</td>
<td>Displays the average size in bytes of the row store pages flushed in the critical phase.</td>
</tr>
<tr>
<td>LAST_RTT_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size in bytes of the rollback transaction table at last savepoint. This may be less than the sum of components if there are duplicate TIDs.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view shows information about executed savepoints, and is associated with the ALTER SYSTEM SAVEPOINT statement.

The `START_TIME`, `STATE`, `VERSION`, and `LAST_*` columns relate to the last executed or currently executing savepoint. Other columns contain aggregated values. Refer to the M_SAVEPOINTS system view for further information about various counters.

This view has a resettable counterpart; you can see the values since the last reset in the M_SAVEPOINT_STATISTICS_RESET system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_SAVEPOINT_STATISTICS_RESET;`.

**Related Information**

- M_SAVEPOINT_STATISTICS_RESET System View [page 2101]
- M_SAVEPOINTS System View [page 2093]
- ALTER SYSTEM SAVEPOINT Statement (System Management) [page 567]
6.2.251 M_SAVEPOINT_STATISTICS_RESET System View

Provides the savepoint statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_SAVEPOINT_STATISTICS. Refer to M_SAVEPOINT_STATISTICS for information about the structure and use of this view.

This view is associated with the ALTER SYSTEM SAVEPOINT statement.

In addition to the members mentioned in M_SAVEPOINT_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_SAVEPOINT_STATISTICS System View [page 2097]
ALTER SYSTEM SAVEPOINT Statement (System Management) [page 567]

6.2.252 M_SCHEDULER_JOBS System View

Shows the status and the history of the status of the scheduled jobs.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the job.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user who will execute, who is executing, or who has executed the job.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the job.</td>
</tr>
<tr>
<td>SCHEDULER_JOB_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the job.</td>
</tr>
<tr>
<td>STATUS</td>
<td>NVARCHAR(16)</td>
<td>Displays the status of the job. Valid statuses are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SCHEDULED if the job is scheduled for execution in the future</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RUNNING if the job is currently executing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SUCCESS if the job executed successfully</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ERROR if an error occurred</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>NVARCHAR(2048)</td>
<td>Displays an error message in the case where STATUS is ERROR.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time when the job is scheduled, if STATUS is SCHEDULED; otherwise, the time when the job execution started.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time when the job execution ended if STATUS is SUCCESS or ERROR; otherwise NULL.</td>
</tr>
</tbody>
</table>

**Additional Information**

Users see jobs that they created or on which they have been granted the ALTER or DROP object privilege. Users with the CATALOG READ system privilege see status and history for all jobs.

**6.2.253 M_SECURESTORE System View**

Provides secure store monitoring information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY_TYPE</td>
<td>VARCHAR(12)</td>
<td>Displays the type of encryption key.</td>
</tr>
<tr>
<td>IS_CONSISTENT</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the key is consistent between the persistence and the SSFS file: TRUE/FALSE.</td>
</tr>
<tr>
<td>RESET_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of consistency resets.</td>
</tr>
<tr>
<td>VERSION</td>
<td>INTEGER</td>
<td>Displays the version of the key.</td>
</tr>
<tr>
<td>IS_CURRENT</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the key version is the current version in the SSFS file: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
6.2.254 M_SEMAPHORES System View

Provides semaphore statistics.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>STATISTICS_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the statistics object name.</td>
</tr>
<tr>
<td>STATISTICS_ID</td>
<td>BIGINT</td>
<td>Displays the statistics object unique ID.</td>
</tr>
<tr>
<td>WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of wait calls.</td>
</tr>
<tr>
<td>BLOCKING_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of blocking wait calls.</td>
</tr>
<tr>
<td>TIMEOUT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of timeouts.</td>
</tr>
<tr>
<td>WAIT_RATE</td>
<td>DOUBLE</td>
<td>Displays the wait rate percentage.</td>
</tr>
<tr>
<td>LAST_BLOCKING_TIME</td>
<td>BIGINT</td>
<td>Displays the last time of blocking wait calls in microseconds.</td>
</tr>
<tr>
<td>MAX_BLOCKING_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of blocking wait calls in microseconds.</td>
</tr>
<tr>
<td>MIN_BLOCKING_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of blocking wait calls in microseconds.</td>
</tr>
<tr>
<td>SUM_BLOCKING_TIME</td>
<td>BIGINT</td>
<td>Displays the total time of blocking wait calls in microseconds.</td>
</tr>
<tr>
<td>AVG_BLOCKING_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of blocking wait calls in microseconds.</td>
</tr>
<tr>
<td>CREATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of semaphore creations (for shared statistics only).</td>
</tr>
<tr>
<td>DESTROY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of semaphore destructions (for shared statistics only).</td>
</tr>
</tbody>
</table>
### Additional Information

This view contains information about single semaphore objects or groups of semaphore objects. It does not contain information about all semaphores.

This view has a resettable counterpart; you can see the values since the last reset in the `M_SEMAPHORES_RESET` system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_SEMAPHORES_RESET;`.

### Related Information

- **M_SEMAPHORES_RESET** System View [page 2104]

### 6.2.255 M_SEMAPHORES_RESET System View

Provides semaphore statistics since the last reset.

This view contains values accumulated since the last reset of the main view `M_SEMAPHORES`. Refer to `M_SEMAPHORES` for information about the structure and use of this view.

In addition to the members mentioned in `M_SEMAPHORES`, this view also contains a timestamp field, `RESET_TIME`, which indicates the last time the data was reset.

### Related Information

- **M_SEMAPHORES** System View [page 2103]
**6.2.256 M_SEQUENCES System View**

Provides statistics for sequence caches.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where the sequence cache exists. This is only displayed when the cache is greater than 1 and the value has been cached.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port where the sequence cache exists. This is only displayed when the cache is greater than 1 and the value has been cached.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the sequence.</td>
</tr>
<tr>
<td>SEQUENCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the sequence name.</td>
</tr>
<tr>
<td>CACHE_SIZE</td>
<td>INTEGER</td>
<td>Displays the cache size of the sequence cache in bytes.</td>
</tr>
<tr>
<td>START_VALUE</td>
<td>DECIMAL</td>
<td>Displays the start value of the sequence cache. This is only displayed when the cache is greater than 1 and the value has been cached.</td>
</tr>
<tr>
<td>END_VALUE</td>
<td>DECIMAL</td>
<td>Displays the end value of the sequence cache. This is only displayed when the cache is greater than 1 and the value has been cached.</td>
</tr>
<tr>
<td>CURRENT_VALUE</td>
<td>DECIMAL</td>
<td>Displays the current value of the sequence cache in bytes.</td>
</tr>
</tbody>
</table>

**Related Information**

[SEQUENCES System View](#) [page 1654]
[CREATE SEQUENCE Statement (Data Definition)](#) [page 810]
[ALTER SEQUENCE Statement (Data Definition)](#) [page 479]
# 6.2.257 M_SERIES_TABLES System View

Provides statistics on the physical contents of series tables.

## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column table name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original</td>
</tr>
<tr>
<td></td>
<td></td>
<td>table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is in progress.</td>
</tr>
<tr>
<td>SERIES_KEY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of distinct series identified by the SERIES KEY columns.</td>
</tr>
<tr>
<td>PERIOD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of distinct periods represented in the period column.</td>
</tr>
<tr>
<td>ROWS_EQUIDISTANT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows optimized to be in a piecewise equidistant seg-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ment.</td>
</tr>
<tr>
<td>ROWS_NOT_EQUIDISTANT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows that have not been optimized to be in a piecewi-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sewise equidistant segment.</td>
</tr>
<tr>
<td>ROWS_REORGANIZED_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows that have been optimized by the ALTER TABLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SERIES REORGANIZE statement.</td>
</tr>
</tbody>
</table>
### Related Information

**ALTER TABLE Statement (Data Definition) [page 589]**

6.2.258 **M_SERVICES System View**

Provides the status of all services.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name. See M_SERVICE_TYPES for all known service names.</td>
</tr>
<tr>
<td>PROCESS_ID</td>
<td>INTEGER</td>
<td>Displays the process ID.</td>
</tr>
<tr>
<td>DETAIL</td>
<td>VARCHAR(128)</td>
<td>Internal use only. Use COORDINATOR_TYPE to test the service role.</td>
</tr>
<tr>
<td>ACTIVE_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the active status: NO, YES, UNKNOWN, STARTING, or STOPPING.</td>
</tr>
<tr>
<td>SQL_PORT</td>
<td>INTEGER</td>
<td>Displays the SQL port.</td>
</tr>
<tr>
<td>COORDINATOR_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the coordinator type in the distributed landscape: MASTER, SLAVE, STANDBY, or NONE.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
IS_DATABASE_LOCAL | VARCHAR(2) | Displays whether or not the service is local to the database, tenant, or if it belongs to one specific database.

**Related Information**

- ALTER SYSTEM STOP SERVICE Statement (System Management) [page 585]
- ALTER SYSTEM RECONFIGURE SERVICE Statement (System Management) [page 556]
- M_SERVICE_TYPES System View [page 2132]

### 6.2.259 M_SERVICE_COMPONENT_DETAILS System View

Allows closer monitoring of service components.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE_NAME</td>
<td>VARCHAR(256)</td>
<td></td>
<td>Displays the database name.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td></td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td></td>
<td>Displays the service name. See M_SERVICE_TYPES for all known service names.</td>
</tr>
<tr>
<td>SERVICE_COMPONENT</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the component name. See M_SERVICE_COMPONENTS for all known component names.</td>
</tr>
<tr>
<td>SERVICE_COMPONENT_ACTION</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the name of the component action.</td>
</tr>
<tr>
<td>PROGRESS_DETAILS</td>
<td>VARCHAR(128)</td>
<td></td>
<td>Displays the status details and progress information.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>START_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td></td>
<td>Displays the timestamp of the start of the component action or lifecycle phase.</td>
</tr>
<tr>
<td>DURATION</td>
<td>BIGINT</td>
<td>Microsecond</td>
<td>Displays the duration of the component action or lifecycle phase. The value is -1 as long as progress is ongoing.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_SERVICES System View [page 2107]
- M_SERVICE_COMPONENT_HISTORY System View [page 2109]
- M_SERVICE_COMPONENT_MEMORY System View [page 2110]
- M_SERVICE_COMPONENTS System View [page 2111]

**6.2.260 M_SERVICE_COMPONENT_HISTORY System View**

Displays persisted information from the M_SERVICE_COMPONENTS and M_SERVICE_COMPONENT_DETAILS system views.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE_NAME</td>
<td>VARCHAR(256)</td>
<td></td>
<td>Displays the database name.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td></td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td></td>
<td>Displays the service name. See M_SERVICE_TYPES for all known service names.</td>
</tr>
<tr>
<td>SERVICE_START_TIME- STAMP</td>
<td>TIMESTAMP</td>
<td></td>
<td>Displays the timestamp of the start of the service.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LIFECYCLE_PHASE</td>
<td>VARCHAR(32)</td>
<td></td>
<td>Displays the lifecycle phase of the component.</td>
</tr>
<tr>
<td>SERVICE_COMPONENT</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the component name. See M_SERVICE_COMPONENTS for all known component names.</td>
</tr>
<tr>
<td>SERVICE_COMPONENT_ACTION</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the name of the component action.</td>
</tr>
<tr>
<td>PROGRESS_DETAILS</td>
<td>VARCHAR(128)</td>
<td></td>
<td>Displays the status details and progress information.</td>
</tr>
<tr>
<td>START_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td></td>
<td>Displays the timestamp of the start of the component action or lifecycle phase.</td>
</tr>
<tr>
<td>DURATION</td>
<td>BIGINT</td>
<td>Microsecond</td>
<td>Displays the duration of the component action or lifecycle phase.</td>
</tr>
</tbody>
</table>

**Related Information**

M_SERVICE_COMPONENT_DETAILS System View [page 2108]
M_SERVICE_COMPONENT_MEMORY System View [page 2110]
M_SERVICE_COMPONENTS System View [page 2111]

### 6.2.261 M_SERVICE_COMPONENT_MEMORY System View

Provides service-specific memory usage by logical component.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where the service is running.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port where the service is running.</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>VARCHAR(64)</td>
<td>Displays the logical component for which memory usage is reported.</td>
</tr>
<tr>
<td>USED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the amount of memory, in bytes, being used for the logical component.</td>
</tr>
</tbody>
</table>

### 6.2.262 M_SERVICE_COMPONENTS System View

Displays all known components and their current status within an SAP HANA service.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE_COMPONENT</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the service component name.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td></td>
<td>Displays the service name. See M_SERVICE_TYPES for all known service names.</td>
</tr>
<tr>
<td>IS_ENABLED</td>
<td>VARCHAR(5)</td>
<td></td>
<td>Displays whether the component is enabled or not: TRUE/FALSE.</td>
</tr>
<tr>
<td>LIFECYCLE_PHASE</td>
<td>VARCHAR(32)</td>
<td></td>
<td>Displays the lifecycle phase of the component.</td>
</tr>
<tr>
<td>START_THREAD_ID</td>
<td>BIGINT</td>
<td></td>
<td>Displays the ID of the thread that started the component.</td>
</tr>
<tr>
<td>START_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td></td>
<td>Displays the timestamp when the component started.</td>
</tr>
<tr>
<td>START_DURATION</td>
<td>BIGINT</td>
<td>Microsecond</td>
<td>Displays the component start duration.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ASSIGN_THREAD_ID</td>
<td>BIGINT</td>
<td></td>
<td>Displays the ID of the thread that assigned the component.</td>
</tr>
<tr>
<td>ASSIGN_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td></td>
<td>Displays the timestamp of the beginning of the component ASSIGN.</td>
</tr>
<tr>
<td>ASSIGN_DURATION</td>
<td>BIGINT</td>
<td>Microsecond</td>
<td>Displays the duration of the ASSIGN component.</td>
</tr>
<tr>
<td>POST_ASSIGN_THREAD_ID</td>
<td>BIGINT</td>
<td></td>
<td>Displays the POST_ASSIGN thread ID of the component.</td>
</tr>
<tr>
<td>POST_ASSIGN_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td></td>
<td>Displays the timestamp of the start of the POST_ASSIGN component.</td>
</tr>
<tr>
<td>POST_ASSIGN_DURATION</td>
<td>BIGINT</td>
<td>Microsecond</td>
<td>Displays the duration of the POST_ASSIGN component.</td>
</tr>
<tr>
<td>SHUTDOWN_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td></td>
<td>Displays the timestamp of the start of the component shutdown.</td>
</tr>
<tr>
<td>SHUTDOWN_DURATION</td>
<td>BIGINT</td>
<td>Microsecond</td>
<td>Displays the duration of the component shutdown.</td>
</tr>
<tr>
<td>LAST_RECONFIG_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td></td>
<td>Displays the timestamp of the start of the last component reconfiguration.</td>
</tr>
<tr>
<td>RECONFIG_DURATION</td>
<td>BIGINT</td>
<td>Microsecond</td>
<td>Displays the duration of the last component reconfiguration.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td></td>
<td>Displays the internal port.</td>
</tr>
</tbody>
</table>

**Additional Information**

Each component displays if it is enabled within the service and only then does it contain further information about the various phases (start, assign, reconfiguration, and shutdown). In case a service was assigned multiple times, only the latest assigned service is visible. The same goes for the other phases. Furthermore, a service is only represented in the view if it has a ComponentManager implemented, which should be the case for all services that are not used for test-only purposes. To give some insight into the accumulated
various component-specific functionality that is not yet extracted into its own components, the monitoring view contains an additional row for each phase, which is marked as an <other> component and adds up these remaining durations.

Another additional row for each phase, which is marked as an <all> component, adds up all durations per phase.

The timestamp and the duration of the <other> component cannot be used to calculate the exact end time of a phase, since it excludes some overhead that happens within the internal phases of a service.

Related Information

M_SERVICE_COMPONENT_MEMORY System View [page 2110]
M_SERVICES System View [page 2107]
M_SERVICE_COMPONENTDETAILS System View [page 2108]
M_SERVICE_COMPONENT_HISTORY System View [page 2109]

6.2.263 M_SERVICE_MEMORY System View

Displays detailed memory utilization information by services.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td></td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td></td>
<td>Displays the service name.</td>
</tr>
<tr>
<td>PROCESS_ID</td>
<td>INTEGER</td>
<td></td>
<td>Displays the process ID.</td>
</tr>
<tr>
<td>LOGICAL_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the virtual memory size, in bytes, from the operating system perspective.</td>
</tr>
<tr>
<td>PHYSICAL_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the physical resident memory size, in bytes, from the operating system perspective.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CODE_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the code size, including shared libraries in bytes.</td>
</tr>
<tr>
<td>STACK_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the stack size in bytes.</td>
</tr>
<tr>
<td>HEAP_MEMORY_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the heap part of the memory pool.</td>
</tr>
<tr>
<td>HEAP_MEMORY_USED_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the amount of pool heap memory that is in use in bytes.</td>
</tr>
<tr>
<td>SHARED_MEMORY_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the shared memory part of the memory pool in bytes.</td>
</tr>
<tr>
<td>SHARED_MEMORY_USED_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the amount of pool shared memory that is in use in bytes.</td>
</tr>
<tr>
<td>COMPACTORS_ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the part of the memory pool that can potentially (if unpinned) be freed during a memory shortage in bytes.</td>
</tr>
<tr>
<td>COMPACTORS_FREEABLE_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the memory that can be freed during a memory shortage in bytes.</td>
</tr>
<tr>
<td>ALLOCATION_LIMIT</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the maximum memory pool size in bytes (configurable value).</td>
</tr>
<tr>
<td>EFFECTIVE_ALLOCATION_LIMIT</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the effective maximum memory pool size, in bytes, considering the pool sizes of other processes (computed value).</td>
</tr>
<tr>
<td>BLOCKED_MEMORY_LIMIT</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the minimum guaranteed memory for the process in bytes.</td>
</tr>
<tr>
<td>FREE_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Byte</td>
<td>Displays the allocated free memory in bytes.</td>
</tr>
</tbody>
</table>
### 6.2.264 M_SERVICE_NETWORK_IO System View

Provides service network I/O statistics.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENDER_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name of the sending service.</td>
</tr>
<tr>
<td>SENDER_PORT</td>
<td>INTEGER</td>
<td>Displays the port that the sending service listens on.</td>
</tr>
<tr>
<td>RECEIVER_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name of the receiving service.</td>
</tr>
</tbody>
</table>
### Additional Information

This view has a resettable counterpart; you can see the values since the last reset in the M_SERVICE_NETWORK_IO_RESET system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_SERVICE_NETWORK_IO_RESET;`.

### Related Information

**M_SERVICE_NETWORK_IO_RESET System View [page 2116]**

#### 6.2.265 M_SERVICE_NETWORK_IO_RESET System View

Provides the service network I/O statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_SERVICE_NETWORK_IO. Refer to M_SERVICE_NETWORK_IO for information about the structure and use of this view.

In addition to the members mentioned in M_SERVICE_NETWORK_IO, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

### Related Information

**M_SERVICE_NETWORK_IO System View [page 2115]**
6.2.266  M_SERVICE_NETWORK_METHOD_IO System View

Displays the number of calls and amount of data that is sent and received.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENDER_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name of the sender.</td>
</tr>
<tr>
<td>SENDER_PORT</td>
<td>INTEGER</td>
<td>Displays the host port number of the sender.</td>
</tr>
<tr>
<td>RECEIVER_HOST</td>
<td>NVARCHAR(256)</td>
<td>Displays the host name of the receiver.</td>
</tr>
<tr>
<td>RECEIVER_PORT</td>
<td>NVARCHAR(256)</td>
<td>Displays the host port number of the receiver.</td>
</tr>
<tr>
<td>THREAD_METHOD</td>
<td>NVARCHAR(256)</td>
<td>Displays the originating method name.</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>NVARCHAR(256)</td>
<td>Displays additional details.</td>
</tr>
<tr>
<td>REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of requests for the method.</td>
</tr>
<tr>
<td>SEND_SIZE</td>
<td>BIGINT</td>
<td>Displays the total number of bytes sent.</td>
</tr>
<tr>
<td>RECEIVE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total number of bytes received.</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last updated timestamp.</td>
</tr>
</tbody>
</table>

Related Information

M_SERVICE_NETWORK_METHOD_IO_RESET System View [page 2118]
6.2.267 M_SERVICE_NETWORK_METHOD_IO_RESET System View

Provides service network method I/O statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_SERVICE_NETWORK_METHOD_IO. Refer to M_SERVICE_NETWORK_METHOD_IO for information about the structure and use of this view.

In addition to the members mentioned in M_SERVICE_NETWORK_METHOD_IO, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_SERVICE_NETWORK_METHOD_IO System View [page 2117]

6.2.268 M_SERVICE_REPLICATION System View

Provides information about replicated services.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER(10)</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID.</td>
</tr>
<tr>
<td>SITE_ID</td>
<td>INTEGER</td>
<td>Displays the generated site ID.</td>
</tr>
<tr>
<td>SITE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the logical site name.</td>
</tr>
<tr>
<td>SECONDARY_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the secondary host name.</td>
</tr>
<tr>
<td>SECONDARY_PORT</td>
<td>INTEGER</td>
<td>Displays the secondary port.</td>
</tr>
<tr>
<td>SECONDARY_SITE_ID</td>
<td>INTEGER</td>
<td>Displays the generated ID of the secondary site.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SECONDARY_SITE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the secondary logical site name.</td>
</tr>
<tr>
<td>SECONDARY_ACTIVE_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the secondary active status.</td>
</tr>
<tr>
<td>SECONDARY_CONNECT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time the connection was established from the secondary site.</td>
</tr>
<tr>
<td>SECONDARY_RECONNECT_COUNT</td>
<td>INTEGER</td>
<td>Displays the secondary reconnect count.</td>
</tr>
<tr>
<td>SECONDARY_FAILOVER_COUNT</td>
<td>INTEGER</td>
<td>Displays the secondary failover count.</td>
</tr>
<tr>
<td>SECONDARY_FULLY_RECOVERABLE</td>
<td>VARCHAR(5)</td>
<td>Displays that the secondary system can be fully recovered with a backup from the primary system. If this value is FALSE, then the backup history is broken. If there is a takeover at that time, then start a new data backup once the takeover is finished.</td>
</tr>
<tr>
<td>REPLICATION_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the replication mode.</td>
</tr>
<tr>
<td>REPLICATION_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the replication status.</td>
</tr>
<tr>
<td>REPLICATION_STATUS_DETAILS</td>
<td>VARCHAR(1.024)</td>
<td>Displays the replication status details.</td>
</tr>
<tr>
<td>FULL_SYNC</td>
<td>VARCHAR(16)</td>
<td>Displays the full sync status:                                                                -------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DISABLED: the full sync is not configured</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ENABLED: the full sync is configured, but it is not yet active (transactions do not block in this state)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ACTIVE: the full sync mode is configured and active</td>
</tr>
<tr>
<td>LAST_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the current log position.</td>
</tr>
<tr>
<td>LAST_LOG_POSITION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the current log position timestamp.</td>
</tr>
<tr>
<td>LAST_SAVEPOINT_VERSION</td>
<td>INTEGER</td>
<td>Displays the current savepoint version.</td>
</tr>
<tr>
<td>LAST_SAVEPOINT_LOGPOSITION</td>
<td>BIGINT(19)</td>
<td>Displays the current savepoint log position.</td>
</tr>
<tr>
<td>LAST_SAVEPOINT_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the current savepoint timestamp.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>SHIPPED_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the shipped log position.</td>
</tr>
<tr>
<td>SHIPPED_LOG_POSITION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the shipped log position time.</td>
</tr>
<tr>
<td>SHIPPED_LOG_BUFFERS_COUNT</td>
<td>BIGINT</td>
<td>Displays the shipped log buffers count.</td>
</tr>
<tr>
<td>SHIPPED_LOG_BUFFERS_SIZE</td>
<td>BIGINT</td>
<td>Displays the shipped log buffers size in bytes.</td>
</tr>
<tr>
<td>SHIPPED_LOG_BUFFERS_DURATION</td>
<td>BIGINT</td>
<td>Displays the shipped log buffer duration in microseconds.</td>
</tr>
<tr>
<td>REPLAYED_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the log end position of the last known replayed log buffer on the secondary site.</td>
</tr>
<tr>
<td>REPLAYED_LOG_POSITION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of the last known replayed log buffer on the secondary site.</td>
</tr>
<tr>
<td>SHIPPED_SAVEPOINT_VERSION</td>
<td>INTEGER</td>
<td>Displays the shipped savepoint version.</td>
</tr>
<tr>
<td>SHIPPED_SAVEPOINT_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the shipped savepoint log position.</td>
</tr>
<tr>
<td>SHIPPED_SAVEPOINT_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the shipped savepoint start time.</td>
</tr>
<tr>
<td>SHIPPED_FULL_REPLICA_COUNT</td>
<td>BIGINT</td>
<td>Displays the shipped full replica count.</td>
</tr>
<tr>
<td>SHIPPED_FULL_REPLICA_SIZE</td>
<td>BIGINT</td>
<td>Displays the shipped full replica size in bytes.</td>
</tr>
<tr>
<td>SHIPPED_FULL_REPLICA_DURATION</td>
<td>BIGINT</td>
<td>Displays the shipped full replica duration in microseconds.</td>
</tr>
<tr>
<td>SHIPPED_LAST_FULL_REPLICA_SIZE</td>
<td>BIGINT</td>
<td>Displays the shipped last full replica size in bytes.</td>
</tr>
<tr>
<td>SHIPPED_LAST_FULL_REPLICA_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the shipped last full replica start time.</td>
</tr>
<tr>
<td>SHIPPED_LAST_FULL_REPLICA_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the shipped last full replica end time.</td>
</tr>
<tr>
<td>SHIPPED_DELTA_REPLICA_COUNT</td>
<td>BIGINT</td>
<td>Displays the shipped delta replica count.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SHIPPED_DELTA_REPLICA_SIZE</td>
<td>BIGINT</td>
<td>Displays the shipped delta replica size in bytes.</td>
</tr>
<tr>
<td>SHIPPED_DELTA_REPLICA_DURATION</td>
<td>BIGINT</td>
<td>Displays the shipped delta replica duration in microseconds.</td>
</tr>
<tr>
<td>SHIPPED_LAST_DELTA_REPLICA_SIZE</td>
<td>BIGINT</td>
<td>Displays the shipped last delta replica size in bytes.</td>
</tr>
<tr>
<td>SHIPPED_LAST_DELTA_REPLICA_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the shipped last delta replica start time.</td>
</tr>
<tr>
<td>SHIPPED_LAST_DELTA_REPLICA_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the shipped last delta replica end time.</td>
</tr>
<tr>
<td>ASYNC_BUFFER_FULL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times the asynchronous replication buffer was full.</td>
</tr>
<tr>
<td>BACKLOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the current replication backlog in bytes.</td>
</tr>
<tr>
<td>MAX_BACKLOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum replication backlog in bytes.</td>
</tr>
<tr>
<td>BACKLOG_TIME</td>
<td>BIGINT</td>
<td>Displays the current replication backlog in microseconds.</td>
</tr>
<tr>
<td>MAX_BACKLOG_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum replication backlog in microseconds.</td>
</tr>
<tr>
<td>REPLAY_BACKLOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the size, in bytes, of all log buffers that have been shipped to the secondary site but have not yet been replayed on the secondary site.</td>
</tr>
<tr>
<td>MAX_REPLAY_BACKLOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size, in bytes, of the REPLAY_BACKLOG_SIZE since the system startup.</td>
</tr>
<tr>
<td>REPLAY_BACKLOG_TIME</td>
<td>BIGINT</td>
<td>Displays the time difference, in microseconds, between the time of the last shipped log buffer and the last replayed log buffer on the secondary site.</td>
</tr>
<tr>
<td>MAX_REPLAY_BACKLOG_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time, in microseconds, of the REPLAY_BACKLOG_TIME since the system startup.</td>
</tr>
</tbody>
</table>
6.2.269 M_SERVICE_STATISTICS System View

Provides statistics on active services.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name.</td>
</tr>
<tr>
<td>PROCESS_ID</td>
<td>INTEGER</td>
<td>Displays the process ID.</td>
</tr>
<tr>
<td>DETAIL</td>
<td>VARCHAR(128)</td>
<td>Displays the detail information, similar to COORDINATOR_TYPE in M_SERVICES.</td>
</tr>
<tr>
<td>ACTIVE_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the active status. This column contains the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• NO - Displays that the service has not started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• YES - Displays that the service has started and is ready for requests.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• UNKNOWN - Displays the initial state after starting a landscape, or when</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the state is not known (for example, after a crash). If the service does</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not start within a minute, this state changes to NO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STARTING - Displays that the service is starting. This state might last</td>
</tr>
<tr>
<td></td>
<td></td>
<td>for several minutes on first startup or recovery.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STOPPING - Displays that the service is stopping.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the process start timestamp.</td>
</tr>
<tr>
<td>SYS_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the current system timestamp.</td>
</tr>
<tr>
<td>PROCESS_CPU</td>
<td>SMALLINT</td>
<td>Displays the CPU usage percentage of the current process. See additional</td>
</tr>
<tr>
<td></td>
<td></td>
<td>notes after this table.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROCESS_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the CPU usage of the current process since the start in milliseconds.</td>
</tr>
<tr>
<td>TOTAL_CPU</td>
<td>SMALLINT</td>
<td>Displays the CPU usage percentage of all processes. See additional notes after this table.</td>
</tr>
<tr>
<td>TOTAL_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the CPU usage of all processes since the start in milliseconds. See additional notes after this table.</td>
</tr>
<tr>
<td>PROCESS_MEMORY</td>
<td>BIGINT</td>
<td>Displays the process logical memory usage in bytes.</td>
</tr>
<tr>
<td>PROCESS_PHYSICAL_MEMORY</td>
<td>BIGINT</td>
<td>Displays the process physical memory usage in bytes.</td>
</tr>
<tr>
<td>TOTAL_MEMORY</td>
<td>BIGINT</td>
<td>Displays the host physical and swap memory usage in bytes.</td>
</tr>
<tr>
<td>AVAILABLE_MEMORY</td>
<td>BIGINT</td>
<td>Displays the host physical and swap memory size in bytes.</td>
</tr>
<tr>
<td>PHYSICAL_MEMORY</td>
<td>BIGINT</td>
<td>Displays the host physical memory size in bytes.</td>
</tr>
<tr>
<td>REQUESTS_PER_SEC</td>
<td>DOUBLE</td>
<td>Displays the requests per second. For internal use only.</td>
</tr>
<tr>
<td>RESPONSE_TIME</td>
<td>INTEGER</td>
<td>Displays the response time in milliseconds. For internal use only.</td>
</tr>
<tr>
<td>FINISHED_NON_INTERNAL_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of finished requests. For internal use only.</td>
</tr>
<tr>
<td>ALL_FINISHED_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the finished requests, including internal requests. For internal use only.</td>
</tr>
<tr>
<td>ACTIVE_REQUEST_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of active requests. For internal use only.</td>
</tr>
<tr>
<td>PENDING_REQUEST_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of pending requests. For internal use only.</td>
</tr>
<tr>
<td>ACTIVE_THREAD_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of active threads.</td>
</tr>
<tr>
<td>THREAD_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of total threads.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>OPEN_FILE_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of open files.</td>
</tr>
</tbody>
</table>

### Additional Information

**PROCESS_CPU, TOTAL_CPU**

PROCESS_CPU and TOTAL_CPU contain CPU usage in percent since last select. This select could be done by another user or another session.

To calculate exact CPU usage use 2 selects and this formula:

\[
\begin{align*}
\text{process_cpu} &= \frac{(\text{select2.process_cpu-time} - \text{select1.process_cpu_time})}{(\text{select2.sys_timestamp} - \text{select1.sys_timestamp})} \\
\text{total_cpu} &= \frac{(\text{select2.total_cpu_time} - \text{select1.total_cpu_time})}{(\text{select2.sys_timestamp} - \text{select1.sys_timestamp})}
\end{align*}
\]

**TOTAL_CPU_TIME** TOTAL_CPU_TIME counts the CPU usage as if there would be 1 core. M_HOST_RESOURCE_UTILIZATION shows similar values multiplied by the number of cores.

```sql
select s.host, s.total_cpu_time, (h.total_cpu_user_time + h.total_cpu_system_time) / i.value
from m_service_statistics s, m_host_resource_utilization h, m_host_information i
where s.service_name = 'nameserver' and i.key = 'cpu_threads' and s.host = h.host
and s.host = i.host
```

### Related Information

**M_HOST_RESOURCE_UTILIZATION** System View [page 1937]

**6.2.270 M_SERVICE_THREADS** System View

Displays detailed information about threads created by services.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name. See M_SERVICE_TYPES for all known service names.</td>
</tr>
<tr>
<td>HIERARCHY</td>
<td>VARCHAR(128)</td>
<td>Displays the thread grouping information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This column contains the connection ID, the update transaction ID, and the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>transaction ID. The column is empty if the threads are inactive.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>THREAD_ID</td>
<td>BIGINT</td>
<td>Displays the thread ID.</td>
</tr>
<tr>
<td>THREAD_TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the thread type.</td>
</tr>
<tr>
<td>THREAD_METHOD</td>
<td>VARCHAR(256)</td>
<td>Displays the thread method.</td>
</tr>
<tr>
<td>THREAD_DETAIL</td>
<td>NCLOB</td>
<td>Displays the thread detail.</td>
</tr>
<tr>
<td>THREAD_STATE</td>
<td>VARCHAR(32)</td>
<td>Displays the thread state.</td>
</tr>
<tr>
<td>IS_ACTIVE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the thread is active (starting, running, stopping, waiting,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>etc.).</td>
</tr>
<tr>
<td>DURATION</td>
<td>BIGINT</td>
<td>Displays the thread duration in milliseconds.</td>
</tr>
<tr>
<td>CALLER</td>
<td>VARCHAR(256)</td>
<td>Displays the service that called the thread.</td>
</tr>
<tr>
<td>CALLING</td>
<td>VARCHAR(256)</td>
<td>Displays the service called by the thread.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the ID of the statement being executed.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the MD5 hash value for STATEMENT_STRING.</td>
</tr>
<tr>
<td>ROOT_STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the ID of the root statement being executed.</td>
</tr>
<tr>
<td>ROOT_STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the MD5 hash value for the root statement string.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the SQL user name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user name.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays that the application can define which source file SAP HANA is called from. This value is also displayed in the M_PREPARED_STATEMENTS.APPLICATION_SOURCE system view.</td>
</tr>
<tr>
<td>CPU_TIME_SELF</td>
<td>BIGINT</td>
<td>Displays the active CPU time of the thread in microseconds.</td>
</tr>
<tr>
<td>CPU_TIME_CUMULATIVE</td>
<td>BIGINT</td>
<td>Displays the active CPU time of the thread and its associated children in microseconds.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP of the client machine.</td>
</tr>
<tr>
<td>CLIENT_PID</td>
<td>BIGINT</td>
<td>Displays the client process ID.</td>
</tr>
<tr>
<td>STATEMENT_EXECUTION_ID</td>
<td>BIGINT</td>
<td>Displays the statement execution ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction ID.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the update transaction ID.</td>
</tr>
<tr>
<td>LOCK_WAIT_COMPONENT</td>
<td>VARCHAR(32)</td>
<td>Displays the component assigned to the lock.</td>
</tr>
<tr>
<td>LOCK_WAIT_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the internal name of the lock (can be joined with the STATISTICS_NAME column from, for example, M_MUTEXES).</td>
</tr>
<tr>
<td>LOCK_OWNER_THREAD_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the thread that is holding the lock.</td>
</tr>
<tr>
<td>LOCKS_OWNED</td>
<td>VARCHAR(256)</td>
<td>Displays the locks currently owned by the thread, including the lock type (shared/exclusive). This column does not show all locks that are monitored.</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>INTEGER</td>
<td>Displays the statement priority.</td>
</tr>
<tr>
<td>STATEMENT_THREAD_LIMIT</td>
<td>INTEGER</td>
<td>Displays the statement thread limit.</td>
</tr>
<tr>
<td>STATEMENT_MEMORY_LIMIT</td>
<td>INTEGER</td>
<td>Displays the statement memory limit.</td>
</tr>
<tr>
<td>PASSPORT_ROOTCONTEXT_ID</td>
<td>VARBINARY(16)</td>
<td>Displays the passport root context ID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td>PASSPORT_TRANSACTION_ID</td>
<td>NVARCHAR(32)</td>
<td>Displays the passport transaction ID.</td>
</tr>
<tr>
<td>PASSPORT_CONNECTION_ID</td>
<td>VARBINARY(16)</td>
<td>Displays the passport connection ID.</td>
</tr>
<tr>
<td>PASSPORT_CONNECTION_COUNTER</td>
<td>BIGINT</td>
<td>Displays the passport connection counter.</td>
</tr>
<tr>
<td>PASSPORT_COMPONENT_NAME</td>
<td>NVARCHAR(32)</td>
<td>Displays the passport component name.</td>
</tr>
<tr>
<td>PASSPORT_ACTION</td>
<td>NVARCHAR(40)</td>
<td>Displays the passport action.</td>
</tr>
<tr>
<td>NUMA_NODE_INDEX</td>
<td>SMALLINT</td>
<td>Displays the last known NUMA node that the thread was executed on.</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the workload class.</td>
</tr>
</tbody>
</table>

**Additional Information**

While this view is publicly visible, its contents can only be seen by users with the DATABASE ADMIN privilege.

**Related Information**

- M_SERVICE_THREAD_CALLSTACKS System View [page 2128]
- M_SERVICE_THREAD_SAMPLES System View [page 2129]

Service Details
Analyzing System Performance
6.2.271 M_SERVICE_THREAD_CALLSTACKS System View

Provides stack frame information for service threads.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name. See M_SERVICE_TYPES for all known service names.</td>
</tr>
<tr>
<td>THREAD_ID</td>
<td>BIGINT</td>
<td>Displays the thread ID.</td>
</tr>
<tr>
<td>IS_ACTIVE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the thread is active.</td>
</tr>
<tr>
<td>FRAME_LEVEL</td>
<td>INTEGER</td>
<td>Displays the level of the stack frame.</td>
</tr>
<tr>
<td>FRAME_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the name of stack frame (function, file, library, and so on).</td>
</tr>
</tbody>
</table>

**Additional Information**

While this view is publicly visible, its contents can only be seen by users with the DATABASE ADMIN privilege.

**Related Information**

M_SERVICE_TYPES System View [page 2132]
M_SERVICE_THREADS System View [page 2124]
M_SERVICE_THREAD_SAMPLES System View [page 2129]
## 6.2.272 M_SERVICE_THREAD_SAMPLES System View

Displays detailed information about locks held by threads.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name. See the M_SERVICE_TYPES system view for all known service names.</td>
</tr>
<tr>
<td>HIERARCHY</td>
<td>VARCHAR(128)</td>
<td>Displays the thread grouping information. This column contains the connection ID, the update transaction ID, and the transaction ID. The column is empty if the threads are inactive.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of the record.</td>
</tr>
<tr>
<td>THREAD_ID</td>
<td>BIGINT</td>
<td>Displays the thread ID.</td>
</tr>
<tr>
<td>THREAD_TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the thread type.</td>
</tr>
<tr>
<td>THREAD_METHOD</td>
<td>VARCHAR(256)</td>
<td>Displays the thread method.</td>
</tr>
<tr>
<td>THREAD_DETAIL</td>
<td>NVARCHAR(256)</td>
<td>Displays the thread detail (truncated).</td>
</tr>
<tr>
<td>THREAD_STATE</td>
<td>VARCHAR(32)</td>
<td>Displays the thread state.</td>
</tr>
<tr>
<td>IS_ACTIVE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the thread is active.</td>
</tr>
<tr>
<td>DURATION</td>
<td>BIGINT</td>
<td>Displays the duration of the operation in milliseconds.</td>
</tr>
<tr>
<td>CALLER</td>
<td>VARCHAR(256)</td>
<td>Displays the service, which called this thread.</td>
</tr>
<tr>
<td>CALLING</td>
<td>VARCHAR(256)</td>
<td>Displays the service, which the thread calls.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the unique identifier for an SQL string.</td>
</tr>
<tr>
<td>ROOT_STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the ID of the root statement being executed.</td>
</tr>
<tr>
<td>ROOT_STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the MD5 hash value for the root statement string.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction ID.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the write transaction ID (this number increment).</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the SQL user name.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user name.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays that the application can define which source file SAP HANA is called from. The usage is up to the application. This value is also displayed in M_PREPARED_STATEMENTS.APPLICATION_SOURCE.</td>
</tr>
<tr>
<td>CPU_TIME_SELF</td>
<td>BIGINT</td>
<td>Displays the active CPU time of the thread in microseconds.</td>
</tr>
<tr>
<td>CPU_TIME_CUMULATIVE</td>
<td>BIGINT</td>
<td>Displays the active CPU time of the thread and any associated children in microseconds.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP of the client machine.</td>
</tr>
<tr>
<td>CLIENT_PID</td>
<td>BIGINT</td>
<td>Displays the client process ID.</td>
</tr>
<tr>
<td>STATEMENT_EXECUTION_ID</td>
<td>BIGINT</td>
<td>Displays the statement execution ID.</td>
</tr>
<tr>
<td>LOCK_WAIT_COMPONENT</td>
<td>VARCHAR(32)</td>
<td>Displays the waiting time for the lock component.</td>
</tr>
<tr>
<td>LOCK_WAIT_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the waiting time for the lock ID (can be joined with STATISTICS_NAME from, for example, M_MUTEXES).</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOCK_OWNER_THREAD_ID</td>
<td>BIGINT</td>
<td>Displays the waiting time for the thread ID.</td>
</tr>
<tr>
<td>LOCKS_OWNED</td>
<td>VARCHAR(256)</td>
<td>Displays the locks currently owned by the thread, including the lock type (shared/exclusive). Not all locks are monitored and show up here.</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>INTEGER</td>
<td>Displays the effective statement priority.</td>
</tr>
<tr>
<td>STATEMENT_THREAD_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective statement thread limit.</td>
</tr>
<tr>
<td>STATEMENT_MEMORY_LIMIT</td>
<td>INTEGER</td>
<td>Displays the effective statement memory limit.</td>
</tr>
<tr>
<td>PASSPORT_ROOTCONTEXT_ID</td>
<td>VARBINARY(16)</td>
<td>Displays the passport root context ID.</td>
</tr>
<tr>
<td>PASSPORT_TRANSACTION_ID</td>
<td>NVARCHAR(32)</td>
<td>Displays the passport transaction ID.</td>
</tr>
<tr>
<td>PASSPORT_CONNECTION_ID</td>
<td>VARBINARY(16)</td>
<td>Displays the passport connection ID.</td>
</tr>
<tr>
<td>PASSPORT_CONNECTION_COUNTER</td>
<td>BIGINT</td>
<td>Displays the passport connection counter.</td>
</tr>
<tr>
<td>PASSPORT_COMPONENT_NAME</td>
<td>NVARCHAR(32)</td>
<td>Displays the passport component name.</td>
</tr>
<tr>
<td>PASSPORT_ACTION</td>
<td>NVARCHAR(40)</td>
<td>Displays the passport action.</td>
</tr>
<tr>
<td>NUMA_NODE_INDEX</td>
<td>SMALLINT</td>
<td>Displays the last known NUMA node that the thread was executed on.</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the workload class.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_PREPARED_STATEMENTS System View [page 2055]
- M_SERVICE_THREADS System View [page 2124]
- M_SERVICE_THREAD_CALLSTACKS System View [page 2128]
- M_SERVICE_TYPES System View [page 2132]
6.2.273  M_SERVICE_TRACES System View

Provides configured trace components for each service type.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name.</td>
</tr>
<tr>
<td>COMPONENT_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the trace component name.</td>
</tr>
</tbody>
</table>

6.2.274  M_SERVICE_TYPES System View

Provides information about service types.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name.</td>
</tr>
<tr>
<td>INIFILE</td>
<td>VARCHAR(36)</td>
<td>Displays the configuration file name of service.</td>
</tr>
<tr>
<td>HAS_DETAIL</td>
<td>VARCHAR(5)</td>
<td>Displays whether the service shows details in M_SERVICE_STATISTICS: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

M_SERVICE_STATISTICS System View [page 2122]
6.2.275 M_SESSIONCONTEXT System View

Displays the session variables set for each connection.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>KEY</td>
<td>VARCHAR(32)</td>
<td>Displays the key name of a session context variable.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(512)</td>
<td>Displays the value of a session context variable.</td>
</tr>
<tr>
<td>SECTION</td>
<td>VARCHAR(8)</td>
<td>Displays the section name to distinguish system and user variables:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• USER=application defined variable using SET command or client API call.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SYSTEM=predefined variable or server property.</td>
</tr>
</tbody>
</table>

Additional Information

Execute the following statement to view the assigned session variables for the current connection:

```
SELECT * FROM M_SESSIONCONTEXT WHERE connection_id=current_connection;
```

Related Information

SET [SESSION] Statement (Session Management) [page 1171]
SESSION_CONTEXT Function (Miscellaneous) [page 341]
Session Variables [page 75]
Session Management Statements [page 1205]
6.2.276 M_SHARED_MEMORY System View

Provides shared memory usage information for the SAP HANA indexserver.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>VARCHAR(128)</td>
<td>Displays the module name.</td>
</tr>
<tr>
<td>ALLOCATED_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated shared memory size on the module in bytes.</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the used shared memory size on the module in bytes.</td>
</tr>
<tr>
<td>FREE_SIZE</td>
<td>BIGINT</td>
<td>Displays the free shared memory size on the module in bytes.</td>
</tr>
</tbody>
</table>

6.2.277 M_SNAPSHOTS System View

Provides information about existing snapshots.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>ID</td>
<td>BIGINT</td>
<td>Displays the snapshot ID.</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the creation time.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PURPOSE</td>
<td>VARCHAR(24)</td>
<td>Displays why the snapshot was executed.</td>
</tr>
<tr>
<td>FOR_BACKUP</td>
<td>VARCHAR(5)</td>
<td>Displays if the snapshot was created for backup: TRUE/FALSE.</td>
</tr>
<tr>
<td>ANCHOR</td>
<td>BIGINT</td>
<td>Displays the anchor.</td>
</tr>
<tr>
<td>REDO_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the redo log position corresponding to the snapshot.</td>
</tr>
<tr>
<td>LAST_COMMIT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp of the last commit of the snapshot.</td>
</tr>
</tbody>
</table>

Related Information

Monitoring and Managing Tenant Databases
M_DATABASES System View [page 1846]

6.2.278 M_SQLSCRIPT_CODE_COVERAGE_OBJECT_DEFINITIONS System View

Provides definitions for the objects referenced in SQLScript code coverage results.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_DEFINITION_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the object definition.</td>
</tr>
<tr>
<td>OBJECT_DEFINITION</td>
<td>NCLOB</td>
<td>Displays the definition of the object.</td>
</tr>
</tbody>
</table>

Related Information

SAP HANA SQLScript Reference
M_SQLSCRIPT_CODE_COVERAGE_RESULTS System View [page 2136]
ALTER SYSTEM [START | STOP] SQLSCRIPT CODE COVERAGE Statement (System Management) [page 580]
6.2.279 M_SQLSCRIPT_CODE_COVERAGE_RESULTS System View

Provides per-session SQLScript code coverage results.

## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_DEFINITION_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the object definition. This value corresponds to a value in M_SQLSCRIPT_CODE_COVERAGE_OBJECT_DEFINITIONS.OBJECT_DEFINITION_ID.</td>
</tr>
<tr>
<td>DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the database.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the object.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of object:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• FUNCTION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PROCEDURE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ANONYMOUS BLOCK</td>
</tr>
<tr>
<td>OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the ID of the object.</td>
</tr>
<tr>
<td>LINE</td>
<td>INTEGER</td>
<td>Displays the line number of the code snippet in the object definition.</td>
</tr>
<tr>
<td>COLUMN</td>
<td>INTEGER</td>
<td>Displays the starting column of the code snippet in the object definition.</td>
</tr>
<tr>
<td>START_POSITION</td>
<td>INTEGER</td>
<td>Displays the index of the code snippet’s first character in the object definition.</td>
</tr>
<tr>
<td>END_POSITION</td>
<td>INTEGER</td>
<td>Displays the index of the code snippet’s last character in the object definition.</td>
</tr>
<tr>
<td>HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the hit count for the code snippet during a single invocation of the object; zero if not covered.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVOKER_OBJECT_DEFINITION_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the invoker’s object definition; this value is NULL if this is the top-most function or procedure.</td>
</tr>
<tr>
<td>INVOKER_LINE</td>
<td>INTEGER</td>
<td>Displays the line number in the invoker’s object definition; this value is NULL if this is the top-most function or procedure.</td>
</tr>
<tr>
<td>INVOKER_COLUMN</td>
<td>INTEGER</td>
<td>Displays the column in the invoker’s object definition; this value is NULL if this is the top-most function or procedure.</td>
</tr>
<tr>
<td>INVOKER_START_POSITION</td>
<td>INTEGER</td>
<td>Displays the index of the first character in the invoker’s object definition; this value is NULL if this is the top-most function or procedure.</td>
</tr>
<tr>
<td>INVOKER_END_POSITION</td>
<td>INTEGER</td>
<td>Displays the index of the last character in the invoker’s object definition; this value is NULL if this is the top-most function or procedure.</td>
</tr>
</tbody>
</table>

**Related Information**

SAP HANA SQLScript Reference

ALTER SYSTEM (START | STOP) SQLSCRIPT CODE COVERAGE Statement (System Management) [page 580]

M_SQLSCRIPT_CODE_COVERAGE_OBJECT_DEFINITIONS System View [page 2135]

6.2.280 M_SQLSCRIPT_PLAN_PROFILERS System View

Lists the sessions and procedures that are currently being profiled by the SQLScript plan profiler.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp that the ALTER SYSTEM ... SQLSCRIPT PLAN PROFILER statement was executed. Displays the ID of the user who executed the ALTER SYSTEM ... SQLSCRIPT PLAN PROFILER statement.</td>
</tr>
<tr>
<td>PROCEDURE_CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the connection ID if a FOR SESSION filter is specified, otherwise NULL.</td>
</tr>
<tr>
<td>PROCEDURE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the procedure schema name if a FOR PROCEDURE filter is specified, otherwise, NULL.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the procedure name if a FOR PROCEDURE filter is specified.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name where profiler is executed.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port where profiler is executed.</td>
</tr>
</tbody>
</table>

### Additional Information

Displays the ID of the user who executed the ALTER SYSTEM command. Below are some examples of how this view can be used, and the values it returns based on whether profiling is enabled:

<table>
<thead>
<tr>
<th>SQL Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER SYSTEM START SQLSCRIPT PLAN PROFILER FOR SESSION 123456; SELECT * FROM M_SQLSCRIPT_PLAN_PROFILERS;</td>
<td>&lt;user&gt;; &lt;time&gt;; 123456; null(?); null(?)</td>
</tr>
<tr>
<td>ALTER SYSTEM STOP SQLSCRIPT PLAN PROFILER FOR SESSION 123456; SELECT * FROM M_SQLSCRIPT_PLAN_PROFILERS;</td>
<td>Returns an empty result because there is no active profiler for the session</td>
</tr>
<tr>
<td>ALTER SYSTEM START SQLSCRIPT PLAN PROFILER FOR PROCEDURE S1.P1; SELECT * FROM M_SQLSCRIPT_PLAN_PROFILERS;</td>
<td>&lt;user&gt;; &lt;time&gt;; null(?); S1; P1</td>
</tr>
<tr>
<td>ALTER SYSTEM STOP SQLSCRIPT PLAN PROFILER; SELECT * FROM M_SQLSCRIPT_PLAN_PROFILERS;</td>
<td>Returns an empty result because there is no active profiler.</td>
</tr>
</tbody>
</table>

### Permissions

To see all entries in the M_SQLSCRIPT_PLAN_PROFILERS system view, you must have the CATALOG READ or OPTIMIZER ADMIN system privilege.
6.2.281 M_SQLSCRIPT_PLAN_PROFILER_RESULTS System View

Stores the results generated by the SQLScript plan profiler.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCEDURE_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the name of host where the outermost procedure started.</td>
</tr>
<tr>
<td>PROCEDURE_PORT</td>
<td>INTEGER</td>
<td>Displays the name of port where the outermost procedure started.</td>
</tr>
<tr>
<td>PROCEDURE_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID that started this outermost procedure.</td>
</tr>
<tr>
<td>PROCEDURE_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the database name of outermost procedure.</td>
</tr>
<tr>
<td>PROCEDURE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of schema of the outermost procedure.</td>
</tr>
<tr>
<td>PROCEDURE_LIBRARY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the library name of the outermost procedure.</td>
</tr>
<tr>
<td>PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the outermost procedure.</td>
</tr>
<tr>
<td>RESULT_ID</td>
<td>INTEGER</td>
<td>Displays the SQLScript profiler ID for the call.</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>VARCHAR(64)</td>
<td>Displays any predefined operations.</td>
</tr>
<tr>
<td>OPERATOR_STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the operator string.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OPERATOR_DETAILS</td>
<td>NCLOB</td>
<td>Displays any key-value pairs of additional information that only belong to specific operations.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start timestamp of the operation.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the end timestamp of the operation.</td>
</tr>
<tr>
<td>DURATION</td>
<td>BIGINT</td>
<td>Displays the clock time, in microseconds, spent in the operation itself.</td>
</tr>
<tr>
<td>ACTIVE_TIME_SELF</td>
<td>BIGINT</td>
<td>Displays the clock time, in microseconds, spent in the operation, excluding child operations.</td>
</tr>
<tr>
<td>ACTIVE_TIME_CUMULATIVE</td>
<td>BIGINT</td>
<td>Displays the clock time, in microseconds, between START_TIME and END_TIME.</td>
</tr>
<tr>
<td>CPU_TIME_SELF</td>
<td>BIGINT</td>
<td>Displays the CPU time, in microseconds, spent in the operation, excluding child operations.</td>
</tr>
<tr>
<td>CPU_TIME_CUMULATIVE</td>
<td>BIGINT</td>
<td>Displays the total CPU time, in microseconds, spent in the operation itself and its children.</td>
</tr>
<tr>
<td>USED_MEMORY_SIZE_SELF</td>
<td>BIGINT</td>
<td>Displays the memory used, in bytes, in the operation itself, excluding its children.</td>
</tr>
<tr>
<td>USED_MEMORY_SIZE_CUMULATIVE</td>
<td>BIGINT</td>
<td>Displays the total memory used, in bytes, in the operation itself, including its children.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID used for the operation.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction ID used for the operation.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the statement ID used for the operation.</td>
</tr>
<tr>
<td>THREAD_ID</td>
<td>BIGINT</td>
<td>Displays the thread ID used for the operation.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OPERATOR_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the database name of the procedure/function where the operator is defined.</td>
</tr>
<tr>
<td>OPERATOR_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the procedure/function where the operator is defined.</td>
</tr>
<tr>
<td>OPERATOR_LIBRARY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the library name of the procedure/function where the operator is defined.</td>
</tr>
<tr>
<td>OPERATOR_PROCEDURE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the procedure/function where the operator is defined.</td>
</tr>
<tr>
<td>OPERATOR_LINE</td>
<td>INTEGER</td>
<td>Displays the SQL line of the operator.</td>
</tr>
<tr>
<td>OPERATOR_COLUMN</td>
<td>INTEGER</td>
<td>Displays the SQL column of the operator.</td>
</tr>
<tr>
<td>OPERATOR_POSITION</td>
<td>INTEGER</td>
<td>Displays the SQL position of the operator.</td>
</tr>
<tr>
<td>OPERATOR_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where the operation occurred.</td>
</tr>
<tr>
<td>OPERATOR_PORT</td>
<td>INTEGER</td>
<td>Displays the port where the operation occurred.</td>
</tr>
<tr>
<td>OPERATOR_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the operation.</td>
</tr>
<tr>
<td>PARENT_OPERATOR_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the parent operation.</td>
</tr>
</tbody>
</table>

**Permissions**

To see all entries in the M_SQLSCRIPT_PLAN_PROFILER_RESULTS system view, you must have the CATALOG READ or OPTIMIZER ADMIN system privilege.

**Related Information**

SAP HANA SQLScript Reference

ALTER SYSTEM {START | STOP | CLEAR} SQLSCRIPT PLAN PROFILER Statement (System Management) [page 576]
6.2.282 M_SQLSCRIPT_VARIABLE_CACHE System view

Provides runtime information about procedures and functions that have variable caches defined.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host address of the node on which the cached data is located.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port number of the node on which the cached data is located.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema of the target object.</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the target object name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the target object type: PROCEDURE/FUNCTION.</td>
</tr>
<tr>
<td>VARIABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the variable to be cached.</td>
</tr>
<tr>
<td>VARIABLE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of variable.</td>
</tr>
<tr>
<td>SQLSCRIPT_PLAN_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the execution plan.</td>
</tr>
<tr>
<td>SQLSCRIPT_OPERATOR_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the execution plan operator.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size of the cached data in bytes.</td>
</tr>
<tr>
<td>LAST_CACHE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the date and time when the latest cached data was generated.</td>
</tr>
</tbody>
</table>

### Related Information

SQLSCRIPT_VARIABLE_CACHE System view [page 1660]
6.2.283 M_SQL_ANALYZER_PLAN_TRACES System View

Displays metadata of .plv files generated by the PlanTracer. This information is provided for PUBLIC users.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>NVARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the user who executed the statement.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema that the SQL plan belongs to. SQL plans are generated in each schema even though the statement string is the same since the query optimizer statistics might be different.</td>
</tr>
<tr>
<td>STATEMENT_EXECUTION_ID</td>
<td>NVARCHAR(32)</td>
<td>Displays the unique statement execution ID.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>NVARCHAR(32)</td>
<td>Displays the statement hash of the query string.</td>
</tr>
<tr>
<td>ABAP_VARCHAR_MODE</td>
<td>NVARCHAR(5)</td>
<td>Displays the ABAP VARCHAR mode.</td>
</tr>
<tr>
<td>EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the statement execution time.</td>
</tr>
<tr>
<td>EXECUTION_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the statement execution start time.</td>
</tr>
<tr>
<td>EXECUTION_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the statement execution end time.</td>
</tr>
<tr>
<td>RESULT_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the result record count.</td>
</tr>
</tbody>
</table>

Related Information

Analyzing Statement Performance
6.2.284  M_SQL_CLIENT_NETWORK_IO System View

Provides the client and server elapsed time as well as message sizes for client network messages.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
</tbody>
</table>
| CLIENT_HOST                | NVARCHAR(256) | Displays the host of the client machine, also found in M_CONNEC-
|                            |               | TIONS.CLI-ENT_HOST.                                              |
| CONNECTION_ID              | INTEGER       | Displays the connection ID of the physical server connection.   |
| MESSAGE_ID                 | BIGINT        | Displays the client message ID on the physical server connection.|
| CLIENT_DURATION            | BIGINT        | Displays the elapsed time on the client side including send, se-
|                            |               | rver execution, and receive.                                    |
| SERVER_DURATION            | BIGINT        | Displays the execution time on the server side. Transport time is
|                            |               | the delta between CLIENT_DURATION and SERVER_DURATION.          |
| SERVER_RECEIVED_TIME       | TIMESTAMP     | Displays the time when the message was received on the server se-
|                            |               | rve and begin of execution.                                     |
| RECEIVED_MESSAGE_SIZE      | BIGINT        | Displays the size of messages received on the server side in by-
|                            |               | tes.                                                            |
| SEND_MESSAGE_SIZE          | BIGINT        | Displays the size of the messages sent to the client side in by-
|                            |               | tes.                                                            |
**Additional Information**

By default, collection of statistics related to client network I/O is controlled by the `sql_client_network_io` parameter in the `indexserver.ini` configuration file.

**Related Information**

**Network Performance Analysis on Transactional Level**

**M_CONNECTIONS System View [page 1785]**

**6.2.285 M_SQL_PLAN_CACHE System View**

Provides statistics for an individual execution plan.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the MD5 hash value for the STATEMENT_STRING column.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name who prepared the plan.</td>
</tr>
<tr>
<td>SESSION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the session user name who prepared the plan. This is the same value appears for USER_NAME in the M_CONNECTIONS.USER monitoring view.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name that the SQL plan belongs to. SQL plans are generated in each schema even though the statement string is the same, since the query optimizer statistics might be different.</td>
</tr>
<tr>
<td>SESSION_PROPERTIES</td>
<td>NVARCHAR(500)</td>
<td>Displays the composed variables from the session context.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether a plan is currently validated (TRUE/FALSE). A plan is invalidated when its corresponding schema objects, such as table and view, are changed. Invalidated plans are recompiled if the same statement is executed; otherwise, they are evicted by another plan when newly compiled.</td>
</tr>
<tr>
<td>LAST_INVALIDATION_REASON</td>
<td>VARCHAR(64)</td>
<td>Displays the reason for the last invalidation.</td>
</tr>
<tr>
<td>IS_INTERNAL</td>
<td>VARCHAR(5)</td>
<td>Displays TRUE if the query plan was initiated from a database internal connection, not from the user nor other servers. For example, when an internal SAP HANA component executes a SQL statement related to the Statistics Service (_SYS_STATISTICS), or runs SQL statements in a background thread that exploit SQL Plan Cache, and so on. Displays FALSE if the query plan was initiated by a client application.</td>
</tr>
<tr>
<td>IS_DISTRIBUTED_EXECUTION</td>
<td>VARCHAR(5)</td>
<td>Displays whether tables are located in multiple nodes or multiple services are involved in plan execution (TRUE/FALSE).</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COMPILATION_OPTIONS</td>
<td>VARCHAR(128)</td>
<td>Displays the compilation options that are used by the SQL Query Optimizer when it generates cached execution plans:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Returns an empty string if the SQL Query Optimizer compiled without any special settings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Returns a STATEMENT HINT option when the SQL Query Optimizer compiles and optimizes with a statement hint</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Returns an ABSTRACT SQL PLAN option when the SQL Query Optimizer compiles with an abstract sql plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Returns a STATEMENT HINT ON INNER STATEMENT option when the SQLScript compiler applies a statement hint to an inner statement of this SQLScript plan.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multiple options can used together to create a list of options, for example: STATEMENT HINT, ABSTRACT SQL PLAN.</td>
</tr>
<tr>
<td>IS_PINNED_PLAN</td>
<td>VARCHAR(5)</td>
<td>Displays whether the plan is explicitly stored by the PlanViz utility (TRUE/ FALSE).</td>
</tr>
<tr>
<td>PINNED_PLAN_ID</td>
<td>VARCHAR(5)</td>
<td>This is deprecated.</td>
</tr>
<tr>
<td>ABAP_VARCHAR_MODE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the ABAP VARCHAR mode is enabled (TRUE/ FALSE). Only ABAP application developers are concerned with this mode, which indicates if it is a NULL terminated string or not.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application name information.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays the application source information.</td>
</tr>
<tr>
<td>ACCESSED_TABLES</td>
<td>VARCHAR(5000)</td>
<td>Displays the relevant base table ID list accessed in the plan. The format is ( ) for a list entry, with entries separated by commas.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ACCESSED_TABLE_NAMES</td>
<td>NCLOB</td>
<td>Displays the relevant base table name list accessed in the plan. The format is ( ) for a list entry, with entries separated by commas.</td>
</tr>
<tr>
<td>ACCESSED_OBJECTS</td>
<td>VARCHAR(5000)</td>
<td>Displays the relevant schema object ID list accessed in the plan. The format is ( ) for a list entry, with entries separated by commas.</td>
</tr>
<tr>
<td>ACCESSED_OBJECT_NAMES</td>
<td>NCLOB</td>
<td>Displays the relevant schema object name list accessed in the plan. The format is ( ) for a list entry, with entries separated by commas.</td>
</tr>
<tr>
<td>TABLE_LOCATIONS</td>
<td>VARCHAR(2000)</td>
<td>Displays all relevant table locations for both original and replicated tables. The table location contains the host, port, database ID, and volume ID, for example 1cdb0m:30240, 0, 3.</td>
</tr>
<tr>
<td>TABLE_TYPES</td>
<td>VARCHAR(128)</td>
<td>Displays whether the plan refers to the column store only, row store only, or mixed. Possible values are: ROW, COLUMN, or ROW, COLUMN.</td>
</tr>
<tr>
<td>EXECUTION_ENGINE</td>
<td>VARCHAR(32)</td>
<td>Displays the list of all involved execution frameworks.</td>
</tr>
<tr>
<td>PLAN_SHARING_TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the plan sharing type: GLOBAL, SESSION LOCAL, or LOGICAL SESSION GLOBAL. See below for definitions.</td>
</tr>
<tr>
<td>OWNER_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection owning the plan.</td>
</tr>
<tr>
<td>PLAN_ID</td>
<td>BIGINT</td>
<td>Displays the logical plan ID, which is a non-negative value.</td>
</tr>
<tr>
<td>PLAN_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size used by the plan in bytes.</td>
</tr>
<tr>
<td>REFERENCE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of statements using the plan. When the number reaches zero, it can be evicted by the victim selection policy of the plan cache.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PARAMETER_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of parameters assigned to the execution.</td>
</tr>
<tr>
<td>UPDATED_TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the updated table for the plan.</td>
</tr>
<tr>
<td>LOGICAL_CONNECTION_VOLUME_ID</td>
<td>BIGINT</td>
<td>Displays the volume ID of the logical connection. This value is 0 if there is no session context defining it as a global plan.</td>
</tr>
<tr>
<td>EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated count of the plan execution.</td>
</tr>
<tr>
<td>EXECUTION_COUNT_BY_ROUTING</td>
<td>BIGINT</td>
<td>Displays the accumulated count of the plan execution by a client-routed connection in the statement routing. This column shows how many times the statement is executed in the routed connection. The routed connection is a physical connection in a part of the logical session.</td>
</tr>
<tr>
<td>TOTAL_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the sum of time of the plan execution in microseconds, including the communication time with clients.</td>
</tr>
<tr>
<td>AVG_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the average time of the plan execution in microseconds, including the communication time with clients.</td>
</tr>
<tr>
<td>MIN_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the minimum time of the plan execution in microseconds, including the communication time with clients.</td>
</tr>
<tr>
<td>MAX_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the maximum time of the plan execution in microseconds, including the communication time with clients.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of time of the plan executions in microseconds.</td>
</tr>
<tr>
<td>AVG_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of the plan execution in microseconds.</td>
</tr>
<tr>
<td>MIN_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of the plan execution in microseconds.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of the plan execution in microseconds.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_OPEN_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of time in microseconds establishing result sets.</td>
</tr>
<tr>
<td>AVG_EXECUTION_OPEN_TIME</td>
<td>BIGINT</td>
<td>Displays the average time that the cursor is open in microseconds.</td>
</tr>
<tr>
<td>MIN_EXECUTION_OPEN_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for cursor open in microseconds.</td>
</tr>
<tr>
<td>MAX_EXECUTION_OPEN_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for cursor open in microseconds.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of time for transferring rows in microseconds.</td>
</tr>
<tr>
<td>AVG_EXECUTION_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for cursor fetch in microseconds.</td>
</tr>
<tr>
<td>MIN_EXECUTION_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for cursor fetch in microseconds.</td>
</tr>
<tr>
<td>MAX_EXECUTION_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for cursor fetch in microseconds.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_CLOSE_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of time for result set cleanup in microseconds.</td>
</tr>
<tr>
<td>AVG_EXECUTION_CLOSE_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for cursor close in microseconds.</td>
</tr>
<tr>
<td>MIN_EXECUTION_CLOSE_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for cursor close in microseconds.</td>
</tr>
<tr>
<td>MAX_EXECUTION_CLOSE_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for cursor close in microseconds.</td>
</tr>
<tr>
<td>TOTAL_METADATA_CACHE_MISS_COUNT</td>
<td>BIGINT</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>TOTAL_TABLE_LOAD_TIME_DURING_PREPARATION</td>
<td>BIGINT</td>
<td>Displays the sum of the table loading time during plan preparation in microseconds.</td>
</tr>
<tr>
<td>AVG_TABLE_LOAD_TIME_DURING_PREPARATION</td>
<td>BIGINT</td>
<td>Displays the average table loading time during plan preparation in microseconds.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>MIN_TABLE_LOAD_TIME_DURING_PREPARATION</td>
<td>BIGINT</td>
<td>Displays the minimum table loading time during plan preparation in microseconds.</td>
</tr>
<tr>
<td>MAX_TABLE_LOAD_TIME_DURING_PREPARATION</td>
<td>BIGINT</td>
<td>Displays the maximum table loading time during plan preparation in microseconds.</td>
</tr>
<tr>
<td>PREPARATION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of plan preparations.</td>
</tr>
<tr>
<td>TOTAL_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of time of the plan preparation in microseconds.</td>
</tr>
<tr>
<td>AVG_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of the plan preparation in microseconds.</td>
</tr>
<tr>
<td>MIN_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of the plan preparation in microseconds.</td>
</tr>
<tr>
<td>MAX_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of the plan preparation in microseconds.</td>
</tr>
<tr>
<td>TOTAL_RESULT_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated number of records during plan execution.</td>
</tr>
<tr>
<td>TOTAL_LOCK_WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated lock wait count for the plan.</td>
</tr>
<tr>
<td>TOTAL_LOCK_WAIT_DURATION</td>
<td>BIGINT</td>
<td>Displays the accumulated lock wait duration for the plan in microseconds.</td>
</tr>
<tr>
<td>LAST_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the last connection ID that executed the plan.</td>
</tr>
<tr>
<td>LAST_EXECUTION_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the last execution timestamp.</td>
</tr>
<tr>
<td>LAST_PREPARATION_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the last preparation timestamp.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the sum of memory peaks of all plan executions, in bytes. In cases where a query was executed in a scale-out system (that is, the query was running distributed across multiple hosts), then TOTAL_EXECUTION_MEMORY_SIZE is the highest peak memory usage across all involved hosts. So, if host A had a peak usage of 2 GB and host B has a peak usage of 3 GB, then MEMORY_SIZE displays 3 GB.</td>
</tr>
<tr>
<td>AVG_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the average memory peak of all plan executions, in bytes.</td>
</tr>
<tr>
<td>MIN_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum memory peak of all plan executions, in bytes.</td>
</tr>
<tr>
<td>MAX_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum memory peak of all plan executions, in bytes.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the total amount of tracked CPU time.</td>
</tr>
<tr>
<td>AVG_EXECUTION_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the average amount of tracked CPU time.</td>
</tr>
<tr>
<td>MIN_EXECUTION_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum amount of tracked CPU time.</td>
</tr>
<tr>
<td>MAX_EXECUTION_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum amount of tracked CPU time.</td>
</tr>
<tr>
<td>AVG_SERVICE_NETWORK_REQUEST_COUNT</td>
<td>REAL</td>
<td>Displays the average count of service network requests during plan execution.</td>
</tr>
<tr>
<td>MAX_SERVICE_NETWORK_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum count of service network requests during plan execution.</td>
</tr>
<tr>
<td>TOTAL_SERVICE_NETWORK_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the total count of service network requests during plan execution.</td>
</tr>
<tr>
<td>AVG_CALLED_THREAD_COUNT</td>
<td>REAL</td>
<td>Displays the average count of called threads used during plan execution.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_CALLED_THREAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum count of called threads used during plan execution.</td>
</tr>
<tr>
<td>TOTAL_CALLED_THREAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the total count of called threads used during plan execution.</td>
</tr>
<tr>
<td>TOTAL_BATCH_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays a total batch count per one execution for cached DML plans.</td>
</tr>
<tr>
<td>AVG_BATCH_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays an average batch count per one execution for cached DML plans.</td>
</tr>
<tr>
<td>MIN_BATCH_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays a minimum batch count per one execution for cached DML plans.</td>
</tr>
<tr>
<td>MAX_BATCH_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays a maximum batch count per one execution for cached DML plans.</td>
</tr>
<tr>
<td>AVG_SERVICE_NETWORK_REQUEST_DURATION</td>
<td>REAL</td>
<td>Displays the average time in microseconds of service network requests during plan execution.</td>
</tr>
<tr>
<td>MAX_SERVICE_NETWORK_REQUEST_DURATION</td>
<td>BIGINT</td>
<td>Displays the maximum time in microseconds of service network requests during plan execution.</td>
</tr>
<tr>
<td>TOTAL_SERVICE_NETWORK_REQUEST_DURATION</td>
<td>BIGINT</td>
<td>Displays the total time in microseconds of service network requests during plan execution.</td>
</tr>
<tr>
<td>AVG_SERVICE_NETWORK_REQUEST_SIZE</td>
<td>REAL</td>
<td>Displays the average data amount in bytes of service network requests during plan execution.</td>
</tr>
<tr>
<td>MAX_SERVICE_NETWORK_REQUEST_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum amount in bytes of service network requests during plan execution.</td>
</tr>
<tr>
<td>TOTAL_SERVICE_NETWORK_REQUEST_SIZE</td>
<td>BIGINT</td>
<td>Displays the total amount in bytes of service network requests during plan execution.</td>
</tr>
<tr>
<td>TOTAL_BUFFER_CACHE_PAGE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of times the NSE page was found in buffer cache memory.</td>
</tr>
<tr>
<td>AVG_BUFFER_CACHE_PAGE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the average number of times page the NSE page was found in buffer cache memory.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MIN_BUFFER_CACHE_PAGE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the minimum number of times the NSE page was found in buffer cache memory.</td>
</tr>
<tr>
<td>MAX_BUFFER_CACHE_PAGE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum number of times the NSE page was found in buffer cache memory.</td>
</tr>
<tr>
<td>TOTAL_BUFFER_CACHE_PAGE_MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of times the NSE page had to be read from disk.</td>
</tr>
<tr>
<td>AVG_BUFFER_CACHE_PAGE_MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the average number of times the NSE page had to be read from disk.</td>
</tr>
<tr>
<td>MIN_BUFFER_CACHE_PAGE_MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the minimum number of times the NSE page had to be read from disk.</td>
</tr>
<tr>
<td>MAX_BUFFER_CACHE_PAGE_MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum number of times the NSE page had to be read from disk.</td>
</tr>
<tr>
<td>TOTAL_BUFFER_CACHE_PINNED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of pinned the NSE page memory out of buffer cache.</td>
</tr>
<tr>
<td>AVG_BUFFER_CACHE_PINNED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of pinned the NSE page memory out of buffer cache.</td>
</tr>
<tr>
<td>MIN_BUFFER_CACHE_PINNED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of pinned the NSE page memory out of buffer cache.</td>
</tr>
<tr>
<td>MAX_BUFFER_CACHE_PINNED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of pinned the NSE page memory out of buffer cache.</td>
</tr>
<tr>
<td>TOTAL_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of NSE pages read from disk into the buffer cache.</td>
</tr>
<tr>
<td>AVG_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of NSE pages read from disk into the buffer cache.</td>
</tr>
<tr>
<td>MIN_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of NSE pages read from disk into the buffer cache.</td>
</tr>
<tr>
<td>MAX_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of NSE pages read from disk into the buffer cache.</td>
</tr>
</tbody>
</table>
Additional Information

- **GLOBAL**: A plan always can be shared with any other connections concurrently. There are no restrictions for this plan sharing type.
- **SESSION LOCAL**: A plan is never shared. This sharing type is more restricted than SESSION EXCLUSIVE GLOBAL. If a connection generates a plan, the plan belongs to the connection. This plan cannot be shared with any other connection and it cannot be also shared with any other cursor even if in the same connection.
- **GLOBAL**: -1 (always).
- **LOGICAL SESSION GLOBAL**: owner connection ID with positive value.
- **SESSION LOCAL**: owner connection ID with positive value (never changed).

The M_SQL_PLAN_CACHE view shows statistics for an individual plan but not all plans. It shows whether a specified plan runs longer than expected or which part of execution is dominant. For each cached plan, this view delivers statistics from execution on distributed configuration as well as technical details such as related object IDs, updated objects and so on.

This view has a resettable counterpart; you can see the values since the last reset in the M_SQL_PLAN_CACHE_RESET system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_SQL_PLAN_CACHE_RESET;`.

Related Information

- [ALTER SYSTEM RESET MONITORING VIEW Statement (System Management)](page 565)
- [M_SQL_PLAN_CACHE_RESET System View](page 2161)
- [M_CONNECTIONS System View](page 1785)
- Monitoring SQL Performance with the SQL Plan Cache
- SQL Plan Cache
- indexserver.ini - xsexecagent.ini

6.2.286 M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS System View

Provides statistics for hosts where plans were executed.
## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>PLAN_ID</td>
<td>BIGINT</td>
<td>Displays the logical plan ID as a non-negative value.</td>
</tr>
<tr>
<td>EXECUTION_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where the plan was executed.</td>
</tr>
<tr>
<td>EXECUTION_PORT</td>
<td>INTEGER</td>
<td>Displays the port where the plan was executed.</td>
</tr>
<tr>
<td>TOTAL_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of the tracked memory consumption in bytes.</td>
</tr>
<tr>
<td>AVG_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of the tracked memory consumption in bytes.</td>
</tr>
<tr>
<td>MIN_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of the tracked memory consumption in bytes.</td>
</tr>
<tr>
<td>MAX_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of the tracked memory consumption in bytes.</td>
</tr>
</tbody>
</table>

## Additional Information

This view has a resettable counterpart; you can see the values since the last reset in the M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS_RESET;
```

## Related Information

- ALTER SYSTEM RESET MONITORING VIEW Statement (System Management) [page 565]
- M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS_RESET System View [page 2157]
6.2.287 M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS_RESET System View

Provides statistics for hosts where plans were executed.

Additional Information

This view contains values accumulated since the last reset of the M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS system view. Refer to the topic for M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_SQL_PLAN_CACHE_EXECUTION_LOCATION_STATISTICS System View [page 2155]

6.2.288 M_SQL_PLAN_CACHE_OVERVIEW System View

Provides overall statistics of evicted and cached plans

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PLAN_CACHE_ENABLED</td>
<td>INTEGER</td>
<td>Displays whether the SQL Plan Cash is enabled (TRUE/FALSE).</td>
</tr>
</tbody>
</table>
|                                |              | ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'system') SET('sql', 'plan_cache_enabled') =  
|                                |              | '<True' or 'False'> WITH RECONFIGURE.                                        |
| STATISTICS_COLLECTION_ENABLED  | VARCHAR(5)   | Displays whether the runtime statistics are being collected (TRUE/FALSE).   |
|                                |              | ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'system') SET('sql', 'plan_cache_statistics_enabled') =  
|                                |              | '<True' or 'False'> WITH RECONFIGURE.                                        |
| PLAN_CACHE_CAPACITY            | BIGINT       | Displays the maximum SQL Plan Cache size in bytes.                         |
|                                |              | ALTER SYSTEM ALTER CONFIGURATION ('indexserver.ini', 'system') SET('sql', 'plan_cache_size') =  
|                                |              | '268435456' WITH RECONFIGURE.                                               |
| CACHED_PLAN_SIZE               | BIGINT       | Displays the total size of SQL Plan Cache in bytes.                        |
| PLAN_CACHE_HIT_RATIO           | REAL         | Displays the SQL Plan Cache hit ratio.                                      |
| PLAN_CACHE_LOOKUP_COUNT        | BIGINT       | Displays the number of plan lookup counts from SQL Plan Cache.             |
| PLAN_CACHE_HIT_COUNT           | BIGINT       | Displays the number of hit counts from SQL Plan Cache.                     |
| EVICTED_PLAN_AVG_CACHE_TIME    | BIGINT       | Displays the average duration in microseconds between plan cache insertion  
<p>|                                |              | and eviction.                                                              |
| EVICTED_PLAN_COUNT             | BIGINT       | Displays the number of evicted plans from SQL Plan Cache.                  |
| EVICTED_PLAN_PREPARATION_COUNT | BIGINT       | Displays the total plan preparation count for evicted plans.               |
| EVICTED_PLAN_EXECUTION_COUNT   | BIGINT       | Displays the total plan execution count for evicted plans.                 |</p>
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVICTED_PLAN_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the total duration in microseconds for plan preparation for all evicted plans. It is the sum of the evicted plan's TOTAL_PREPARATION_TIME.</td>
</tr>
<tr>
<td>EVICTED_PLAN_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the total cursor duration in microseconds for evicted plans. It is the sum of the evicted plan's TOTAL_CURSOR_DURATION.</td>
</tr>
<tr>
<td>EVICTED_PLAN_TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total execution time in microseconds for evicted plans. It is the sum of the evicted plan's TOTAL_EXECUTION_TIME.</td>
</tr>
<tr>
<td>EVICTED_PLAN_SIZE</td>
<td>BIGINT</td>
<td>Displays the accumulated total size of evicted plans in bytes.</td>
</tr>
<tr>
<td>CACHED_PLAN_COUNT</td>
<td>BIGINT</td>
<td>Displays the total cached plan count in SQL Plan Cache.</td>
</tr>
<tr>
<td>CACHED_PLAN_PREPARATION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total plan preparation count for cached plans.</td>
</tr>
<tr>
<td>CACHED_PLAN_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total execution count for cached plans.</td>
</tr>
<tr>
<td>CACHED_PLAN_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the total plan preparation duration for cached plans in microseconds.</td>
</tr>
<tr>
<td>CACHED_PLAN_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the total cursor duration for cached plans in microseconds.</td>
</tr>
<tr>
<td>CACHED_PLAN_TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the total execution time for cached plans in microseconds.</td>
</tr>
<tr>
<td>CLEAR_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when SQL Plan Cache was cleared for the last time using ALTER SYSTEM CLEAR SQL PLAN CACHE.</td>
</tr>
</tbody>
</table>

### Additional Information

M_SQL_PLAN_CACHE_OVERVIEW shows the overall statistics of evicted and cached plans. It shows information such as how many times plan eviction occurred and how long it takes to execute currently cached
plans. You can check the eviction rate with the EVICTED_PLAN_COUNT column. If EVICTED_PLAN_COUNT is too high, it means SQL plan cache capacity is not enough and too many compilations occurs.

In addition to statistics, the current status and flags of the SQL plan cache are shown in this view.

**Related Information**

ALTER SYSTEM RESET MONITORING VIEW Statement (System Management) [page 565]
M_SQL_PLAN_CACHE_RESET System View [page 2161]
M_CONNECTIONS System View [page 1785]
Monitoring SQL Performance with the SQL Plan Cache
SQL Plan Cache

### 6.2.289 M_SQL_PLAN_CACHE_PARAMETERS System View

Provides bind parameters for statements cached in SQL Plan Cache. It saves a parameter set of the most expensive execution for each plan.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAN_ID</td>
<td>BIGINT</td>
<td>Displays the logical plan ID which is a non-negative value, same with the one in M_SQL_PLAN_CACHE.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the data type of the parameter.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the position of the parameter in SQL statement.</td>
</tr>
<tr>
<td>PARAMETER_VALUE</td>
<td>NCLOB</td>
<td>Displays the value of the captured parameter.</td>
</tr>
<tr>
<td>BATCH_EXECUTION_ORDER</td>
<td>INTEGER</td>
<td>Displays the execution order in a batch processing.</td>
</tr>
<tr>
<td>EXECUTION_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the parameter value is captured.</td>
</tr>
</tbody>
</table>
Related Information

M_SQL_PLAN_CACHE System View [page 2145]

6.2.290 M_SQL_PLAN_CACHE_RESET System View

Provides statistics of an individual execution plan since the last reset.

This view contains values accumulated since the last reset of the main view M_SQL_PLAN_CACHE. Refer to M_SQL_PLAN_CACHE for information about the structure and use of this view.

In addition to the members mentioned in M_SQL_PLAN_CACHE, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_SQL_PLAN_CACHE System View [page 2145]

6.2.291 M_SQL_PLAN_STATISTICS System View

Provides statistics of a live or evicted individual execution plan.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the MD5 hash value for STATEMENT_STRING.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name who prepared the plan.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SESSION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the SESSION_USER name who prepared the plan. The same value as M_CONNECTIONS.USER_NAME.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name that the SQL plan belongs to.</td>
</tr>
<tr>
<td>SESSION_PROPERTIES</td>
<td>NVARCHAR(500)</td>
<td>Displays the composed variables that can be changed by session context.</td>
</tr>
<tr>
<td>IS_VALID</td>
<td>VARCHAR(5)</td>
<td>Displays whether the plan is currently valid: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_EVICTED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the plan has already been evicted: TRUE/FALSE.</td>
</tr>
<tr>
<td>LAST_INVALIDATION_REASON</td>
<td>VARCHAR(64)</td>
<td>Displays the last plan invalidation reason.</td>
</tr>
<tr>
<td>LAST_EVICTION_REASON</td>
<td>VARCHAR(32)</td>
<td>Displays the last plan eviction reason.</td>
</tr>
<tr>
<td>IS_INTERNAL</td>
<td>VARCHAR(5)</td>
<td>Displays whether the plan is executed from a database internal connection (TRUE) or from a remote connection (FALSE).</td>
</tr>
<tr>
<td>IS_DISTRIBUTED_EXECUTION</td>
<td>VARCHAR(5)</td>
<td>Displays whether the tables are located in multiple nodes: TRUE/FALSE.</td>
</tr>
<tr>
<td>COMPILATION_OPTIONS</td>
<td>VARCHAR(128)</td>
<td>Displays the compilation options that are used by the SQL Query Optimizer when it generates cached execution plans:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Returns an empty string if the SQL Query Optimizer compiled without any special settings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Returns a STATEMENT HINT option when the SQL Query Optimizer compiles and optimizes with a statement hint</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Returns an ABSTRACT SQL PLAN option when the SQL Query Optimizer compiles with an abstract sql plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multiple options can be used together to create a list of options, for example: STATEMENT HINT, ABSTRACT SQL PLAN.</td>
</tr>
<tr>
<td>IS_PINNED_PLAN</td>
<td>VARCHAR(5)</td>
<td>Displays whether the plan is explicitly stored by PlanViz utility: TRUE/FALSE.</td>
</tr>
<tr>
<td>PINNED_PLAN_ID</td>
<td>BIGINT</td>
<td>If this plan is pinned, displays the same value as PINNED_SQL_PLANS.ID.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ABAP_VARCHAR_MODE</td>
<td>VARCHAR(5)</td>
<td>Displays whether ABAP VARCHAR mode is enabled: TRUE/FALSE.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application name information.</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays the application source information.</td>
</tr>
<tr>
<td>ACCESSED_TABLES</td>
<td>VARCHAR(5000)</td>
<td>Displays the relevant base table ID list accessed in the plan.</td>
</tr>
<tr>
<td>ACCESSED_TABLE_NAMES</td>
<td>NCLOB</td>
<td>Displays the relevant base table name list accessed in the plan.</td>
</tr>
<tr>
<td>ACCESSED_OBJECTS</td>
<td>VARCHAR(5000)</td>
<td>Displays the relevant table object ID list accessed in the plan.</td>
</tr>
<tr>
<td>ACCESSED_OBJECT_NAMES</td>
<td>NCLOB</td>
<td>Displays the relevant schema object name list accessed in the plan.</td>
</tr>
<tr>
<td>TABLE_LOCATIONS</td>
<td>VARCHAR(2000)</td>
<td>Displays the relevant table locations for the plan.</td>
</tr>
<tr>
<td>TABLE_TYPES</td>
<td>VARCHAR(128)</td>
<td>Displays whether the plan refers Column store only, Row store only or mixed.</td>
</tr>
<tr>
<td>EXECUTION_ENGINE</td>
<td>VARCHAR(32)</td>
<td>Displays all involved execution frameworks.</td>
</tr>
<tr>
<td>PLAN_SHARING_TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the plan sharing type.</td>
</tr>
<tr>
<td>OWNER_CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the connection owning the plan. The possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• GLOBAL: -1 (always)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SESSION EXCLUSIVE GLOBAL: -1, owner connection ID, which is a positive value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SESSION LOCAL: owner connection ID, which is a positive value (never changed)</td>
</tr>
<tr>
<td>PLAN_ID</td>
<td>BIGINT</td>
<td>Displays the logical plan ID, which is a non-negative value. If the plan is evicted, -1 is shown.</td>
</tr>
<tr>
<td>PLAN_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size used by the plan in bytes.</td>
</tr>
<tr>
<td>EVICTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of evictions that have occurred for the plan.</td>
</tr>
<tr>
<td>LAST_EVICTION_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the plan was evicted for the last time. Returns zero if the plan has never been evicted.</td>
</tr>
<tr>
<td>REFERENCE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of statements using the plan.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PARAMETER_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of parameters to be assigned for the execution.</td>
</tr>
<tr>
<td>UPDATED_TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table changed by the plan.</td>
</tr>
<tr>
<td>LOGICAL_CONNECTION_VOLUME_ID</td>
<td>BIGINT</td>
<td>Displays the volume ID of the logical connection. This value is 0 if there is no session context.</td>
</tr>
<tr>
<td>EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated count of plan execution for valid and evicted plans.</td>
</tr>
<tr>
<td>EXECUTION_COUNT_BY_ROUTING</td>
<td>BIGINT</td>
<td>Displays the accumulated count of plan execution by client-routed connection in statement routing for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the sum of time, in microseconds, that cursors were opened. This includes plan execution and communication time with clients for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the average time, in microseconds, of the plan execution, including communication time with clients for valid and evicted plans.</td>
</tr>
<tr>
<td>MIN_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the minimum time, in microseconds, of the plan execution, including communication time with clients for valid and evicted plans.</td>
</tr>
<tr>
<td>MAX_CURSOR_DURATION</td>
<td>BIGINT</td>
<td>Displays the maximum time, in microseconds, of the plan execution, including communication time with clients for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of time, in microseconds, of the plan execution for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the average time of the plan execution.</td>
</tr>
<tr>
<td>MIN_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time of the plan execution.</td>
</tr>
<tr>
<td>MAX_EXECUTION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time of the plan execution.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_OPEN_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of time in microseconds for cursor open for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_EXECUTION_OPEN_TIME</td>
<td>BIGINT</td>
<td>Displays the average time in microseconds for cursor open for valid and evicted plans.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>MIN_EXECUTION_OPEN_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time in microseconds for cursor open for valid and evicted plans.</td>
</tr>
<tr>
<td>MAX_EXECUTION_OPEN_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time in microseconds for cursor open for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of time in microseconds for cursor fetch for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_EXECUTION_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the average time in microseconds for cursor fetch for valid and evicted plans.</td>
</tr>
<tr>
<td>MIN_EXECUTION_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time in microseconds for cursor fetch for valid and evicted plans.</td>
</tr>
<tr>
<td>MAX_EXECUTION_FETCH_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time in microseconds for cursor fetch for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_CLOSE_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of time in microseconds for cursor close for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_EXECUTION_CLOSE_TIME</td>
<td>BIGINT</td>
<td>Displays the average time in microseconds for cursor close for valid and evicted plans.</td>
</tr>
<tr>
<td>MIN_EXECUTION_CLOSE_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time in microseconds for cursor close for valid and evicted plans.</td>
</tr>
<tr>
<td>MAX_EXECUTION_CLOSE_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time in microseconds for cursor close for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_METADATA_CACHE_MISS_COUNT</td>
<td>BIGINT</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>TOTAL_TABLE_LOAD_TIME_DURING_PREPARATION</td>
<td>BIGINT</td>
<td>Displays the sum of table loading time during plan preparation for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_TABLE_LOAD_TIME_DURING_PREPARATION</td>
<td>BIGINT</td>
<td>Displays the average table loading time in microseconds during plan preparation for valid and evicted plans.</td>
</tr>
<tr>
<td>MIN_TABLE_LOAD_TIME_DURING_PREPARATION</td>
<td>BIGINT</td>
<td>Displays the minimum table loading time during plan preparation for valid and evicted plans.</td>
</tr>
<tr>
<td>MAX_TABLE_LOAD_TIME_DURING_PREPARATION</td>
<td>BIGINT</td>
<td>Displays the maximum table loading time in microseconds during plan preparation for valid and evicted plans.</td>
</tr>
<tr>
<td>PREPARATION_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of plan preparations for valid and evicted plans.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TOTAL_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the sum of time in microseconds of plan preparation for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the average time in microseconds of plan preparation for valid and evicted plans.</td>
</tr>
<tr>
<td>MIN_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time in microseconds of plan preparation for valid and evicted plans.</td>
</tr>
<tr>
<td>MAX_PREPARATION_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time in microseconds of plan preparation for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_RESULT_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated number of records during plan execution for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_LOCK_WAIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the accumulated lock wait count for the plan for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_LOCK_WAIT_DURATION</td>
<td>BIGINT</td>
<td>Displays the accumulated lock wait duration in microseconds for the plan for valid and evicted plans.</td>
</tr>
<tr>
<td>LAST_CONNECTION_ID</td>
<td>BIGINT</td>
<td>Displays the last connection ID that executed the plan.</td>
</tr>
<tr>
<td>LAST_EXECUTION_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the plan was executed for the last time.</td>
</tr>
<tr>
<td>LAST_PREPARATION_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the plan was compiled for the last time.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>With memory tracking, displays the average size in bytes of tracked actual memory consumption for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>With memory tracking, displays the maximum size in bytes of tracked actual memory consumption for valid and evicted plans.</td>
</tr>
<tr>
<td>MIN_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>With memory tracking, displays the minimum size in bytes of tracked actual memory consumption for valid and evicted plans.</td>
</tr>
<tr>
<td>MAX_EXECUTION_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>With memory tracking, displays the total size in bytes of tracked actual memory consumption for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_EXECUTION_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the total amount of tracked CPU time.</td>
</tr>
<tr>
<td>AVG_EXECUTION_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the average amount of tracked CPU time.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MIN_EXECUTION_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum amount of tracked CPU time.</td>
</tr>
<tr>
<td>MAX_EXECUTION_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum amount of tracked CPU time.</td>
</tr>
<tr>
<td>TOTAL_SERVICE_NETWORK_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the total count of service network requests during plan execution for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_SERVICE_NETWORK_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the average count of service network requests during plan execution for valid and evicted plans.</td>
</tr>
<tr>
<td>MAX_SERVICE_NETWORK_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum count of service network requests during plan execution for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_CALLED_THREAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the total count of called threads used during plan execution for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_CALLED_THREAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the average count of called threads used during plan execution for valid and evicted plans.</td>
</tr>
<tr>
<td>MAX_CALLED_THREAD_COUNT</td>
<td>BIGINT</td>
<td>Displays the average count of called threads used during plan execution for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_BATCH_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays a total batch count per one execution for cached DML plans for valid and evicted plans.</td>
</tr>
<tr>
<td>AVG_BATCH_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays an average batch count per one execution for cached DML plans for valid and evicted plans.</td>
</tr>
<tr>
<td>MIN_BATCH_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays a minimum batch count per one execution for cached DML plans for valid and evicted plans.</td>
</tr>
<tr>
<td>MAX_BATCH_EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays a maximum batch count per one execution for cached DML plans for valid and evicted plans.</td>
</tr>
<tr>
<td>TOTAL_SERVICE_NETWORK_REQUEST_DURATION</td>
<td>BIGINT</td>
<td>Displays the total time of the service network requests during plan execution.</td>
</tr>
<tr>
<td>AVG_SERVICE_NETWORK_REQUEST_DURATION</td>
<td>REAL</td>
<td>Displays the average time of the service network requests during plan execution.</td>
</tr>
<tr>
<td>MAX_SERVICE_NETWORK_REQUEST_DURATION</td>
<td>BIGINT</td>
<td>Displays the maximum time of the service network requests during plan execution.</td>
</tr>
<tr>
<td>TOTAL_SERVICE_NETWORK_REQUEST_SIZE</td>
<td>BIGINT</td>
<td>Displays the total amount of service network requests during plan execution in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AVG_SERVICE_NETWORK_REQUEST_SIZE</td>
<td>REAL</td>
<td>Displays the average data amount of service network requests during plan execution in bytes.</td>
</tr>
<tr>
<td>MAX_SERVICE_NETWORK_REQUEST_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum amount of service network requests during plan execution in bytes.</td>
</tr>
<tr>
<td>TOTAL_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of NSE pages read from disk into the buffer cache.</td>
</tr>
<tr>
<td>AVG_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of NSE pages read from disk into the buffer cache.</td>
</tr>
<tr>
<td>MIN_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of NSE pages read from disk into the buffer cache.</td>
</tr>
<tr>
<td>MAX_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of NSE pages read from disk into the buffer cache.</td>
</tr>
<tr>
<td>TOTAL_BUFFER_CACHE_PAGE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of times NSE page was found in buffer cache memory.</td>
</tr>
<tr>
<td>AVG_BUFFER_CACHE_PAGE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the average number of times page NSE page was found in buffer cache memory.</td>
</tr>
<tr>
<td>MIN_BUFFER_CACHE_PAGE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the minimum number of times NSE page was found in buffer cache memory.</td>
</tr>
<tr>
<td>MAX_BUFFER_CACHE_PAGE_HIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum number of times NSE page was found in buffer cache memory.</td>
</tr>
<tr>
<td>TOTAL_BUFFER_CACHE_PAGE_MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of times NSE page had to be read from disk.</td>
</tr>
<tr>
<td>AVG_BUFFER_CACHE_PAGE_MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the average number of times NSE page had to be read from disk.</td>
</tr>
<tr>
<td>MIN_BUFFER_CACHE_PAGE_MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the minimum number of times NSE page had to be read from disk.</td>
</tr>
<tr>
<td>MAX_BUFFER_CACHE_PAGE_MISS_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum number of times NSE page had to be read from disk.</td>
</tr>
<tr>
<td>TOTAL_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of NSE pages had to be read from disk.</td>
</tr>
<tr>
<td>AVG_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of NSE pages had to be read from disk.</td>
</tr>
<tr>
<td>MIN_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of NSE pages had to be read from disk.</td>
</tr>
<tr>
<td>MAX_BUFFER_CACHE_IO_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of NSE pages had to be read from disk.</td>
</tr>
<tr>
<td>TOTAL_BUFFER_CACHE_PINNED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of pinned NSE page memory out of buffer cache.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
AVG_BUFFER_CACHE_PINNED_MEMORY_SIZE | BIGINT | Displays the average size of pinned NSE page memory out of buffer cache.  
MIN_BUFFER_CACHE_PINNED_MEMORY_SIZE | BIGINT | Displays the minimum size of pinned NSE page memory out of buffer cache.  
MAX_BUFFER_CACHE_PINNED_MEMORY_SIZE | BIGINT | Displays the maximum size of pinned NSE page memory out of buffer cache.

### Additional Information

The `M_SQL_PLAN_STATISTICS` view shows statistics for an individual plan, whether it is live or evicted. A plan is evicted from the plan cache, and is not stored in the `M_SQL_PLAN_CACHE` System View, if it is the least recently used cache entry. It shows whether a specified plan runs longer than expected or which part of execution is dominant. For each cached plan, this view delivers statistics from execution on distributed configuration, as well as technical details such as related object IDs, updated objects, and so on.

Information about live and evicted plans is useful in the following scenarios:

- If a previously evicted plan entry is recreated in the plan cache with a high execution statistics number, then you might assume that the plan should not have been evicted and should increase your plan cache size.
- If you want to have an overview of plan statistics, even though some plans have already been evicted.
- If you want to determine which kinds of queries are repeatedly evicted.
- If you want to determine the cause of performance regression, then you can check the statistics of evicted plans for information on how frequently the plan was executed, for how long, and so on.

### Related Information

- `M_SQL_PLAN_CACHE` System View [page 2145]  
- `M_CONNECTIONS` System View [page 1785]  
- `M_SQL_PLAN_STATISTICS_RESET` System View [page 2169]

Monitoring SQL Performance with the SQL Plan Cache

SQL Plan Cache

indexserver.ini - xsexecagent.ini

### 6.2.292 `M_SQL_PLAN_STATISTICS_RESET` System View

Provides statistics of a live or evicted individual execution plan since the last reset.

This view contains values accumulated since the last reset of the main view `M_SQL_PLAN_STATISTICS`. Refer to `M_SQL_PLAN_STATISTICS` for information about the structure and use of this view.
In addition to the members mentioned in M_SQL_PLAN_STATISTICS, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

**Related Information**

M_SQL_PLAN_STATISTICS System View [page 2161]

### 6.2.293 M_STATISTICS_LASTVALUES System View

Provides an in-memory version of _SYS_STATISTICS.STATISTICS_LASTVALUES.

#### Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port name.</td>
</tr>
<tr>
<td>NAME</td>
<td>NVARCHAR(128)</td>
<td>Displays the value name.</td>
</tr>
<tr>
<td>INDEX</td>
<td>NVARCHAR(1024)</td>
<td>Displays the index of value.</td>
</tr>
<tr>
<td>REACHED_AT</td>
<td>TIMESTAMP</td>
<td>Displays the value reached at.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the last value.</td>
</tr>
</tbody>
</table>

#### Additional Information

The Statistics Server when running has a view _SYS_STATISTICS.STATISTICS_LASTVALUES on which this view is based.

**Note**

The new Embedded Statistics Service (ESS) does not use the view _SYS_STATISTICS.STATISTICS_LASTVALUES, which means this view is therefore empty.

**Related Information**

LAST_VALUE Function (Aggregate) [page 238]
6.2.294 M_STREAMING_APPLICATIONS System View

Provides streaming application monitoring information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema in which the application is deployed.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the streaming application.</td>
</tr>
<tr>
<td>APPLICATION_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the application type: PROJECT, HA_PROJECT, or TOOLKIT_ADAPTER.</td>
</tr>
<tr>
<td>APPLICATION_INSTANCE</td>
<td>INTEGER</td>
<td>Displays the instance number of the streaming application.</td>
</tr>
<tr>
<td>APPLICATION_INSTANCE_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of instances that are running or that are started.</td>
</tr>
<tr>
<td>SERVER_ALIAS</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the service on which the application is running.</td>
</tr>
<tr>
<td>REQUESTED_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the status the application is requested to be in by the client: STOPPED/STARTED_RUNNING.</td>
</tr>
<tr>
<td>CURRENT_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the status that the application is currently in:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• START</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STARTING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STARTED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STARTED_RUNNING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STOPPING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STOPPED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STOPPED_FAILED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STOPPED_FAILED_START</td>
</tr>
<tr>
<td>IS_FAILOVER_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the project should be started on another node if the project or node fails: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FAILURE_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of failures that have occurred in the failure interval.</td>
</tr>
<tr>
<td>FAILURE_INTERVAL</td>
<td>INTEGER</td>
<td>Displays the failure interval, in seconds, in which the application is shut down if the failure count reaches the max failures per interval.</td>
</tr>
<tr>
<td>MAX_FAILURES_PER_INTERVAL</td>
<td>INTEGER</td>
<td>Displays the number of failures that can occur in the specified failure interval before the application is shutdown.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_STREAMING_PROJECT_STREAMS System View [page 2180]
- M_STREAMING_SCHEMAS System View [page 2185]
- Developing Applications with SQLScript

**6.2.295 M_STREAMING_PROJECTS System View**

Provides streaming project monitoring information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema in which the application is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name given to the project.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td>Displays the instance number of the project.</td>
</tr>
<tr>
<td>CPU_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total CPU usage, as a percentage, by the project since the last update. These values range from 0 to 100.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CPU_SYSTEM_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total system CPU usage, as a percentage, by the project since the last update. These values range from 0 to 100.</td>
</tr>
<tr>
<td>CPU_USER_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total user CPU usage, as a percentage, by the project since the last update. These values range from 0 to 100.</td>
</tr>
<tr>
<td>TOTAL_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the total CPU time, in microseconds, used by the project since it started.</td>
</tr>
<tr>
<td>TOTAL_CPU_SYSTEM_TIME</td>
<td>BIGINT</td>
<td>Displays the total system CPU time, in microseconds, used by the project since it started.</td>
</tr>
<tr>
<td>TOTAL_CPU_USER_TIME</td>
<td>BIGINT</td>
<td>Displays the total user CPU time, in microseconds, used by the project since it started.</td>
</tr>
<tr>
<td>PROJECT_DURATION</td>
<td>BIGINT</td>
<td>Displays the number of microseconds since the project started.</td>
</tr>
<tr>
<td>VIRTUAL_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the total amount of virtual memory used by the project in bytes.</td>
</tr>
<tr>
<td>RESIDENT_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the total amount of resident memory used by the project in bytes.</td>
</tr>
<tr>
<td>THREAD_COUNT</td>
<td>INTEGER</td>
<td>Displays the total number of threads used by the project at the time of the update.</td>
</tr>
<tr>
<td>MONITORING_INTERVAL</td>
<td>INTEGER</td>
<td>Displays how often stream/client statistics are gathered in seconds.</td>
</tr>
<tr>
<td>IS_ACTIVE_ACTIVE_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Indicates if the Active/Active (Read Enabled) high availability mode is enabled for the project: TRUE/FALSE.</td>
</tr>
<tr>
<td>ACTIVE_ACTIVE_ROLE</td>
<td>VARCHAR(16)</td>
<td>Indicates whether the project instance plays the primary or secondary role in the Active/Active (Read Enabled) high availability mode.</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time that the statistics were updated for the project.</td>
</tr>
</tbody>
</table>
Related Information

M_STREAMING_PROJECT_ADAPTERS System View [page 2174]
M_STREAMING_PROJECT_ADAPTER_STATISTICS System View [page 2176]
M_STREAMING_PROJECT_GD_SUBSCRIPTIONS System View [page 2177]
M_STREAMING_PROJECT_PUBLISHERS System View [page 2178]
M_STREAMING_PROJECT_STREAMS System View [page 2180]
M_STREAMING_PROJECT_SUBSCRIBERS System View [page 2182]

6.2.296 M_STREAMING_PROJECT_ADAPTERS System View

Provides streaming project adapter monitoring information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema in which the application is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name given to the project.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td>Displays the instance number of the project.</td>
</tr>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the adapter.</td>
</tr>
<tr>
<td>ADAPTER_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the adapter type defined in the ATTACH ADAPTER statement.</td>
</tr>
<tr>
<td>STREAM_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the stream on which the adapter is defined.</td>
</tr>
<tr>
<td>IS_INPUT</td>
<td>VARCHAR(5)</td>
<td>Indicates whether or not the connection is an input or an output. This value is TRUE for an input connection and FALSE for an output connection.</td>
</tr>
<tr>
<td>STARTUP_GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the start up group to which the connector belongs.</td>
</tr>
<tr>
<td>ADAPTER_STATE</td>
<td>VARCHAR(32)</td>
<td>Displays the state of the adapter: READY, INITIAL, CONTINOUS, IDLE, DONE, or DEAD.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TOTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of data records recognized in the input data.</td>
</tr>
<tr>
<td>SUCCESSFUL_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of data records successfully processed.</td>
</tr>
<tr>
<td>UNSUCCESSFUL_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of data records that experienced errors.</td>
</tr>
<tr>
<td>MEMORY_USAGE</td>
<td>BIGINT</td>
<td>Displays the total memory used by the adapter, in bytes.</td>
</tr>
<tr>
<td>LAST_ERROR_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time that an error occurred.</td>
</tr>
<tr>
<td>LAST_ERROR_MESSAGE</td>
<td>NVARCHAR(256)</td>
<td>Displays the complete text of the error message as written to the log.</td>
</tr>
<tr>
<td>LATENCY</td>
<td>BIGINT</td>
<td>Displays the latency introduced by the adapter.</td>
</tr>
</tbody>
</table>

**Related Information**

- `M_STREAMING_PROJECT_STREAMS System View [Streaming Analytics] [page 2324]`
- Create an SAP HANA Streaming Analytics Remote Source
- `M_STREAMING_PROJECT_ADAPTER_STATISTICS System View [page 2176]`
- `M_STREAMING_PROJECT_GD_SUBSCRIPTIONS System View [page 2177]`
- `M_STREAMING_PROJECT_PUBLISHERS System View [page 2178]`
- `M_STREAMING_PROJECT_STREAMS System View [page 2180]`
- `M_STREAMING_PROJECT_SUBSCRIBERS System View [page 2182]`
6.2.297 M_STREAMING_PROJECT_ADAPTER_STATISTICS
System View

Provides streaming project adapter statistics monitoring information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema in which the application is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name given to the project.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td>Displays the instance number of the project.</td>
</tr>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the adapter.</td>
</tr>
<tr>
<td>STATISTICS_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the name of the adapter statistic.</td>
</tr>
<tr>
<td>STATISTICS_VALUE</td>
<td>NVARCHAR(256)</td>
<td>Displays the value for the statistic. This value is converted to a string.</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time the statistic was updated for the adapter.</td>
</tr>
</tbody>
</table>

Related Information

M_STREAMING_PROJECT_ADAPTERS System View [page 2174]
M_STREAMING_PROJECT_GD_SUBSCRIPTIONS System View [page 2177]
M_STREAMING_PROJECT_PUBLISHERS System View [page 2178]
M_STREAMING_PROJECT_STREAMS System View [page 2180]
M_STREAMING_PROJECT_SUBSCRIBERS System View [page 2182]
6.2.298 M_STREAMING_PROJECT_GD_SUBSCRIPTIONS
System View

Provides streaming project Guaranteed Delivery (GD) subscription monitoring information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema in which the application is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name given to the project.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td>Displays the instance number of the project.</td>
</tr>
<tr>
<td>GD_KEY</td>
<td>NVARCHAR(1024)</td>
<td>Displays the automatically generated key for the GD session.</td>
</tr>
<tr>
<td>STREAM_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the stream being subscribed to.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the subscribing user.</td>
</tr>
<tr>
<td>GD_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the guaranteed delivery session.</td>
</tr>
<tr>
<td>GD_SEQUENCE_NUMBER</td>
<td>BIGINT</td>
<td>Displays the sequence number of the last event committed from stream_name in this guaranteed delivery session. A value of 0 indicates that no commits have been issued.</td>
</tr>
<tr>
<td>STREAMING_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the unique handle for active external subscribers. For active internal adapters it is always 0. Inactive subscribers have a value of -1.</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time the statistic was updated for the GD subscription.</td>
</tr>
</tbody>
</table>

Related Information

M_STREAMING_PROJECT_ADAPTERS System View [page 2174]
6.2.299 M_STREAMING_PROJECT_PUBLISHERS System View

Provides streaming project publishers monitoring information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema in which the application is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name given to the project.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td>Displays the instance number of the project.</td>
</tr>
<tr>
<td>STREAMING_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the unique handle identifying the connection.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name provided by the client during the connection establish-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ment. This value is shown once the user is authenticated.</td>
</tr>
<tr>
<td>CLIENT_HOST</td>
<td>NVARCHAR(256)</td>
<td>Displays the symbolic host name of the client machine, if available. If not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>available, the host is the IP address of the client machine.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP address of the client machine as a string.</td>
</tr>
<tr>
<td>CLIENT_PORT</td>
<td>INTEGER</td>
<td>Displays the TCP port number from which the connection originates.</td>
</tr>
<tr>
<td>CONNECTION_TAG</td>
<td>NVARCHAR(256)</td>
<td>Displays the user-provided symbolic connection tag name. If this is not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>by the user, then the connection tag is NULL.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TRANSACTION_THROUGHPUT</td>
<td>REAL</td>
<td>Displays the client’s performance, in transactions per second, sent by the client since the last update. Envelopes and any service messages count as transactions.</td>
</tr>
<tr>
<td>ROW_THROUGHPUT</td>
<td>REAL</td>
<td>Displays the client’s performance, in data rows per second, sent by the client since the last update.</td>
</tr>
<tr>
<td>INCREMENTAL_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of transactions, envelopes, or messages sent by the client since the last update.</td>
</tr>
<tr>
<td>INCREMENTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of data rows sent by the client since the last update.</td>
</tr>
<tr>
<td>TOTAL_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of transactions, envelopes, or messages sent by the client.</td>
</tr>
<tr>
<td>TOTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of data rows sent by the client.</td>
</tr>
<tr>
<td>CPU_CORE_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total CPU usage for the client thread, as a percentage of a single CPU core.</td>
</tr>
<tr>
<td>CPU_SYSTEM_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total system CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>CPU_USER_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total user CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>TOTAL_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the total CPU time, in microseconds, since the creation of the client.</td>
</tr>
<tr>
<td>TOTAL_CPU_SYSTEM_TIME</td>
<td>BIGINT</td>
<td>Displays the total system CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>TOTAL_CPU_USER_TIME</td>
<td>BIGINT</td>
<td>Displays the total user CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CONNECT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time the server accepts (but does not authenticate) the connection.</td>
</tr>
<tr>
<td>PUBLISHER_DURATION</td>
<td>BIGINT</td>
<td>Displays the duration of lapsed real time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time the statistics were updated for the publisher.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_STREAMING_PROJECT_ADAPTERS System View [page 2174]
- M_STREAMING_PROJECT_STREAMS System View [page 2180]
- M_STREAMING_PROJECT_SUBSCRIBERS System View [page 2182]
- M_STREAMING_PROJECT_GD_SUBSCRIPTIONS System View [page 2177]

### 6.2.300 M_STREAMING_PROJECT_STREAMS System View

Provides streaming project stream monitoring information.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema in which the application is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name given to the project.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td>Displays the instance number of the project.</td>
</tr>
<tr>
<td>STREAM_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the stream.</td>
</tr>
<tr>
<td>STREAM_TYPE</td>
<td>NVARCHAR(32)</td>
<td>Displays the type of stream: STREAM, DELTASTREAM, WINDOW, or META-DATA.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VISIBILITY</td>
<td>NVARCHAR(32)</td>
<td>Displays the visibility of the stream: INPUT, OUTPUT, LOCAL, or INTERMEDIATE.</td>
</tr>
<tr>
<td>TARGET_NAME</td>
<td>NVARCHAR(256)</td>
<td>Denotes that this stream is an intermediate stream used to populate this target.</td>
</tr>
<tr>
<td>GD_SUPPORT_LEVEL</td>
<td>NVARCHAR(32)</td>
<td>Indicates whether GD (guaranteed delivery) is enabled for the stream or window: NOT_SUPPORTED, SUPPORTED, or SUPPORTED_WITH_CHECKPOINT.</td>
</tr>
<tr>
<td>TRANSACTION_THROUGHPUT</td>
<td>REAL</td>
<td>Displays the stream's performance, in transactions per second, since the last update.</td>
</tr>
<tr>
<td>ROW_THROUGHPUT</td>
<td>REAL</td>
<td>Displays the stream's performance, in rows per second, since the last update.</td>
</tr>
<tr>
<td>INCREMENTAL_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of transactions processed by the server since the last update.</td>
</tr>
<tr>
<td>INCREMENTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of rows processed by the server since the last update.</td>
</tr>
<tr>
<td>QUEUE_SIZE</td>
<td>INTEGER</td>
<td>Displays the stream's input queue size in bytes.</td>
</tr>
<tr>
<td>STORE_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the Number of rows in the store. This is only applicable for Windows.</td>
</tr>
<tr>
<td>SEQUENCE_NUMBER</td>
<td>BIGINT</td>
<td>Displays a sequentially increasing number that represents how many times the statistics have been updated.</td>
</tr>
<tr>
<td>CPU_CORE_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total CPU usage for the stream thread, as a percentage of a single CPU core.</td>
</tr>
<tr>
<td>CPU_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total CPU usage for the stream, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>CPU_SYSTEM_UTILIZATION</td>
<td>REAL</td>
<td>Displays the system CPU usage for the stream, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CPU_USER_UTILIZATION</td>
<td>REAL</td>
<td>Displays the user CPU usage for the stream, as a percentage of all CPU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cores on the machine.</td>
</tr>
<tr>
<td>TOTAL_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the total CPU time since the creation of the stream, in microseconds.</td>
</tr>
<tr>
<td>TOTAL_CPU_SYSTEM_TIME</td>
<td>BIGINT</td>
<td>Displays the total system CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>TOTAL_CPU_USER_TIME</td>
<td>BIGINT</td>
<td>Displays the total user CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time the statistics were updated for the stream.</td>
</tr>
</tbody>
</table>

**Related Information**

- **M_STREAMING_PROJECT_ADAPTERS System View** [page 2174]
- **M_STREAMING_PROJECT_SUBSCRIBERS System View** [page 2182]
- **M_STREAMING_PROJECT_GD_SUBSCRIPTIONS System View** [page 2177]
- **M_STREAMING_PROJECT_PUBLISHERS System View** [page 2178]

### 6.2.301 M_STREAMING_PROJECT_SUBSCRIBERS System View

Provides monitoring information regarding streaming project subscribers.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema in which the application is deployed.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name given to the project.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td>Displays the instance number of the project.</td>
</tr>
<tr>
<td>STREAMING_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the unique handle identifying the connection.</td>
</tr>
<tr>
<td>STREAM_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the stream.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name provided by the client during the connection establish-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ment. This value is shown once the user is authenticated.</td>
</tr>
<tr>
<td>CLIENT_HOST</td>
<td>NVARCHAR(256)</td>
<td>Displays the symbolic host name of the client machine, if available. If not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>available, the host is the IP address of the client machine.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP address of the client machine as a string.</td>
</tr>
<tr>
<td>CLIENT_PORT</td>
<td>INTEGER</td>
<td>Displays the TCP port number from which the connection originates.</td>
</tr>
<tr>
<td>CONNECTION_TAG</td>
<td>NVARCHAR(256)</td>
<td>Displays the user-provided symbolic connection tag name. If this is not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>by the user, then the connection tag is NULL.</td>
</tr>
<tr>
<td>IS_SUBSCRIBED</td>
<td>NVARCHAR(5)</td>
<td>Displays the whether or not the user is subscribed to a stream: TRUE/FALSE.</td>
</tr>
<tr>
<td>TRANSACTION_THROUGHPUT</td>
<td>REAL</td>
<td>Displays the client’s performance, in transactions per second, sent by the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>client since the last update. Envelopes and any service messages count as</td>
</tr>
<tr>
<td></td>
<td></td>
<td>transactions.</td>
</tr>
<tr>
<td>ROW THROUGHPUT</td>
<td>REAL</td>
<td>Displays the client’s performance, in data rows per second, sent by the cli-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ent since the last update.</td>
</tr>
<tr>
<td>INCREMENTAL_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of transactions, envelopes, or messages sent by the cli-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ent since the last update.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INCREMENTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of data rows sent by the client since the last update.</td>
</tr>
<tr>
<td>TOTAL_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of transactions, envelopes, or messages sent by the client.</td>
</tr>
<tr>
<td>TOTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of data rows received by the client.</td>
</tr>
<tr>
<td>TOTAL_ROWS_DROPPED_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of data rows dropped in the gateway because they were not read quickly enough by the client.</td>
</tr>
<tr>
<td>ACCUMULATOR_SIZE</td>
<td>INTEGER</td>
<td>Displays the number of rows collected in the accumulator to be sent in the next pulse. This value is for pulsed subscriptions only.</td>
</tr>
<tr>
<td>QUEUE_SIZE</td>
<td>INTEGER</td>
<td>Displays the number of rows queued for transmission to the client.</td>
</tr>
<tr>
<td>QUEUE_UTILIZATION</td>
<td>REAL</td>
<td>Displays the percentage usage of the queue for transmitting rows to the client.</td>
</tr>
<tr>
<td>WORK_QUEUE_SIZE</td>
<td>INTEGER</td>
<td>Displays the number of rows queued for transmission to the client. These rows have been transferred from the internal queue to the socket buffer and can be regrouped by envelopes.</td>
</tr>
<tr>
<td>CPU_CORE_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total CPU usage for the client thread, as a percentage of a single CPU core.</td>
</tr>
<tr>
<td>CPU_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total CPU usage for the client, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>CPU_SYSTEM_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total systemn CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>CPU_USER_UTILIZATION</td>
<td>REAL</td>
<td>Displays the total user CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
</tbody>
</table>
### Related Information

- [M_STREAMING_PROJECT_GD_SUBSCRIPTIONS System View](#) [page 2177]
- [M_STREAMING_PROJECTS System View](#) [page 2172]

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the schema where streaming applications are deployed.</td>
</tr>
</tbody>
</table>
Related Information

SCHEMAS System View [page 1651]
CREATE SCHEMA Statement (Data Definition) [page 807]
SET SCHEMA Statement (Session Management) [page 1170]
RENAME SCHEMA Statement (Data Definition) [page 1092]
DROP SCHEMA Statement (Data Definition) [page 971]

6.2.303 M_STREAMING_SERVICES System View

Provides streaming service monitoring information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE_ALIAS</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the SAP HANA streaming node.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the name of the host.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
</tbody>
</table>

Related Information

Service Details
M_SERVICES System View [page 2107]
ALTER SYSTEM RECONFIGURE SERVICE Statement (System Management) [page 556]
ALTER SYSTEM STOP SERVICE Statement (System Management) [page 585]
Add Services in a Tenant Database
6.2.304 M_SYSTEM_AVAILABILITY System View

Monitors the availability of the system.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time that this event was originally traced.</td>
</tr>
<tr>
<td>GUID</td>
<td>NVARCHAR (36)</td>
<td>Displays the event guide.</td>
</tr>
<tr>
<td>IS_ORIGIN</td>
<td>VARCHAR (5)</td>
<td>Displays the original entry.</td>
</tr>
<tr>
<td>TRACE_HOST</td>
<td>VARCHAR (64)</td>
<td>Displays the host the trace file was read from.</td>
</tr>
<tr>
<td>EVENT_NAME</td>
<td>VARCHAR (32)</td>
<td>Displays the event name:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• database_add</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• database_remove</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• failover_begin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• failover_end</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• host_remove_prepare</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• host_remove_reorg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• host_remove_abort</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• host_remove</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ping</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• recovery_begin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• recovery_end</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• service_remove</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• service_remove_abort</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• service_remove_prepare</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• service_remove_reorg</td>
</tr>
<tr>
<td>EVENT_DETAIL</td>
<td>VARCHAR (256)</td>
<td>Displays any additional information.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>VARCHAR (256)</td>
<td>Displays the error message.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SYSTEM_ACTIVE</td>
<td>VARCHAR (16)</td>
<td>Displays the system active status: NO, YES, UNKNOWN, STARTING, or STOPPING.</td>
</tr>
<tr>
<td>SYSTEM_STATUS</td>
<td>VARCHAR (16)</td>
<td>Displays the system status: ERROR, IGNORE, INFO, OK, WARNING.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR (64)</td>
<td>Displays the host that traced the event.</td>
</tr>
<tr>
<td>HOST_ACTIVE</td>
<td>VARCHAR (16)</td>
<td>Displays the host active status.</td>
</tr>
<tr>
<td>HOST_STATUS</td>
<td>VARCHAR (16)</td>
<td>Displays the host status.</td>
</tr>
<tr>
<td>DATABASE_NAME</td>
<td>VARCHAR (256)</td>
<td>Displays the database name.</td>
</tr>
<tr>
<td>DATABASE_ACTIVE</td>
<td>VARCHAR (16)</td>
<td>Displays the database active status.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR (32)</td>
<td>Displays the service name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the service port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID.</td>
</tr>
<tr>
<td>SERVICE_ACTIVE</td>
<td>VARCHAR (16)</td>
<td>Displays the service active status.</td>
</tr>
<tr>
<td>HOST_CONFIG_ROLES</td>
<td>VARCHAR (64)</td>
<td>Displays the configured host roles.</td>
</tr>
<tr>
<td>HOST_ACTUAL_ROLES</td>
<td>VARCHAR (64)</td>
<td>Displays the actual host roles.</td>
</tr>
<tr>
<td>STORAGE_CONFIG_PARTITION</td>
<td>INTEGER</td>
<td>Displays the configured storage partition.</td>
</tr>
<tr>
<td>STORAGE_ACTUAL_PARTITION</td>
<td>INTEGER</td>
<td>Displays the actual storage partition.</td>
</tr>
<tr>
<td>TARGET_HOST</td>
<td>VARCHAR (64)</td>
<td>Displays the failover target host.</td>
</tr>
<tr>
<td>TARGET_HOST_CONFIG_ROLES</td>
<td>VARCHAR (64)</td>
<td>Displays the target host configuration roles.</td>
</tr>
<tr>
<td>TARGET_HOST_ACTUAL_ROLES</td>
<td>VARCHAR (64)</td>
<td>Displays the target host actual roles.</td>
</tr>
<tr>
<td>TARGET_STORAGE_CONFIG_PARTITION</td>
<td>INTEGER</td>
<td>Displays the target storage configuration partition.</td>
</tr>
</tbody>
</table>
### Related Information

**Multiple-Host (Distributed) Systems**  
**The System Database**  
**M_SYSTEM_DATA_STATISTICS System View [page 2189]**  
**M_SYSTEM_INFORMATION_STATEMENTS System View [page 2192]**  
**M_SYSTEM_OVERVIEW System View [page 2193]**

### 6.2.305 M_SYSTEM_DATA_STATISTICS System View

Lists data statistics generated by the server when you query a column and row store object.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_STATISTICS_TYPE</td>
<td>VARCHAR(12)</td>
<td>Displays the data statistics object type.</td>
</tr>
<tr>
<td>DATA_SOURCE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the data source object.</td>
</tr>
<tr>
<td>DATA_SOURCE_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the data source object.</td>
</tr>
<tr>
<td>DATA_SOURCE_COLUMN_NAMES</td>
<td>NVARCHAR(5000)</td>
<td>Lists the column names of the data source.</td>
</tr>
<tr>
<td>DATA_SOURCE_STORAGE_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the source storage type of the data source.</td>
</tr>
<tr>
<td>DATA_SOURCE_PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID of the data source.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LAST_REFRESH_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time the data statistics object was generated.</td>
</tr>
</tbody>
</table>
| DATA_STATISTICS_CONTENT         | NCLOB     | Lists the content and properties of the data statistics object as of the last refresh (JSON format). The content section is empty for SIMPLE; it contains values and frequencies in a list for TOPK. For SAMPLE, it contains a list of sample output. The content is similar to exported data for data statistics objects. The properties section contains properties relevant to the data statistics of the specified type. The properties section, when present, may include some of the following properties:  

**BUCKETS**  
Displays the number of buckets in the data statistics object  
**COUNT** Displays the number of rows in the data source  
**DISTINCT COUNT** Displays the number of unique values in the data source  
**NULL COUNT** Displays the number of NULL values in the data source  
**MIN VALUE** Displays the minimum value in the data source in bytes  
**MAX VALUE** Displays the maximum value in the data source in bytes  
**MIN MAX IS VALID** Displays whether MIN/MAX values are valid  
**SAMPLE SIZE** Displays the sample size at build time, expressed as a number |
Additional Information

To select from this view, you must specify an equal predicate on the DATA_SOURCE_OBJECT_NAME and DATA_SOURCE_SCHEMA_NAME columns.

For example:

```
SELECT * FROM M_SYSTEM_DATA_STATISTICS WHERE DATA_SOURCE_SCHEMA_NAME='USER_1'
and DATA_SOURCE_OBJECT_NAME='TABLE1';
```

Predicates on all other columns are optional.

If you use an equal predicate on DATA_SOURCE_COLUMN_NAMES, you can specify one or more column names on a specified table. Separate multiple column names by a comma with no spaces.

If no columns are specified, then TOPK and SIMPLE data statistics for all columns of the specified table are returned, unless the DATA_STATISTICS_TYPE is restricted.

If a single column is specified, then TOPK, SIMPLE, and SAMPLE data statistics for the specified column are returned.

If multiple columns are specified, then only a single SAMPLE data statistics object per partition is returned, containing sample data statistics on all of the specified columns.

For example, this statement returns the server generated SAMPLE statistics on columns COL1, COL2, and COL3 for table USER_1.TABLE1.

```
SELECT * FROM M_SYSTEM_DATA_STATISTICS WHERE DATA_SOURCE_SCHEMA_NAME='USER_1'
and DATA_SOURCE_OBJECT_NAME='TABLE1' AND
DATA_SOURCE_COLUMN_NAMES='COL1,COL2,COL3';
```

Related Information

CREATE STATISTICS Statement (Data Definition) [page 815]
ALTER STATISTICS Statement (Data Definition) [page 484]
REFRESH STATISTICS Statement (Data Definition) [page 1082]
DROP STATISTICS Statement (Data Definition) [page 974]
M_DATA_STATISTICS System View [page 1851]
DATA_STATISTICS System View [page 1538]
6.2.306  M_SYSTEM_INFORMATION_STATEMENTS System View

Provides system information statements.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the statement name.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>NVARCHAR(256)</td>
<td>Displays the statement description.</td>
</tr>
<tr>
<td>STATEMENT</td>
<td>NCLOB</td>
<td>Displays the actual SQL statement.</td>
</tr>
</tbody>
</table>

Related Information

The System Database
System Privileges (Reference)
System Views

6.2.307  M_SYSTEM_LIMITS System View

Provides system limit information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATEGORY</td>
<td>VARCHAR(128)</td>
<td>Displays whether the category of the system limit is adaptable.</td>
</tr>
<tr>
<td>NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the system limit.</td>
</tr>
<tr>
<td>VALUE</td>
<td>VARCHAR(256)</td>
<td>Displays the value of the system limit.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the type of value.</td>
</tr>
</tbody>
</table>
### Related Information

System Limitations [page 1208]
Analyzing System Performance

#### 6.2.308 M_SYSTEM_OVERVIEW System View

Provides an overview of system status including important resource usage information and alerts.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECTION</td>
<td>VARCHAR(16)</td>
<td>Displays the section name.</td>
</tr>
<tr>
<td>NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the key name in section.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(8)</td>
<td>Displays the status value: OK, ERROR, or WARNING. This row is empty for info items.</td>
</tr>
<tr>
<td>VALUE</td>
<td>VARCHAR(256)</td>
<td>Displays the key value in section.</td>
</tr>
</tbody>
</table>

### Related Information

SAP HANA System Architecture Overview
The System Database
System Privileges
System Administration
System Limitations [page 1208]
6.2.309 M_SYSTEM_REPLICATION System View

Monitors system replication information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SITE_ID</td>
<td>INTEGER</td>
<td>Displays the generated site ID.</td>
</tr>
<tr>
<td>SITE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the site name.</td>
</tr>
<tr>
<td>SECONDARY_SITE_ID</td>
<td>INTEGER</td>
<td>Displays the generated site ID of the secondary site.</td>
</tr>
<tr>
<td>SECONDARY_SITE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the secondary logical site name.</td>
</tr>
<tr>
<td>REPLICATION_MODE</td>
<td>VARCHAR(7)</td>
<td>Displays the configured replication mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SYNC: the synchronous replication that acknowledges when a buffer has been written to a disk.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SYNCMEM: the synchronous replication that acknowledges when a buffer has arrived in the memory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ASYNC: the asynchronous replication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• UNKNOWN: set if the replication mode cannot be determined (for example, if there is a communication error when getting status information from a service).</td>
</tr>
<tr>
<td>REPLICATION_STATUS</td>
<td>VARCHAR(12)</td>
<td>Displays the replication status.</td>
</tr>
<tr>
<td>OPERATION_MODE</td>
<td>VARCHAR(32)</td>
<td>Displays the operation mode.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SECONDARY_READ_ACCESS_STATUS</td>
<td>VARCHAR(16)</td>
<td>Indicates whether the secondary system is read-enabled and if read access is activated:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• NOT CONFIGURED: Displays that an operation mode is used that does not allow read access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• STOPPED: Displays that the secondary is running in operation mode logreplay_readaccess, but it is currently stopped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• VERSION MISMATCH: Displays that the secondary system is running in operation mode logreplay_readaccess but read access is internally disabled on the secondary system because it is on a different SAP HANA version than the primary system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• INITIALIZING: Displays that the secondary site is running in operation mode logreplay_readaccess but read access is not yet completely initialized. SQL connections to the secondary system fail in this state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CONSISTENT: Displays that log replay on the secondary site has reached a global consistent state, but the SQL port is not open.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ACTIVE: Displays that the secondary system is running in operation mode logreplay_readaccess and is initialized for read access. SQL connections are possible in this state.</td>
</tr>
</tbody>
</table>

| TIER                        | INTEGER       | Displays the tier on which the source site is located.                                                                                                                                                    |

**Related Information**

ALTER SYSTEM {REGISTER | UNREGISTER} SYSTEM REPLICATION SITE Statement (System Management) [page 559]
ALTER SYSTEM {ENABLE | DISABLE} SYSTEM REPLICATION Statement (System Management) [page 537]
6.2.310 M_SYSTEM_REPLICATION_MVCC_HISTORY System View

Displays the global multi-version concurrency control (MVCC) timestamp history in the secondary site for system replication. The global MVCC timestamp of the secondary site is updated after a chunk of logs from the primary site is replayed on the secondary site.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOBAL_MVCC_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the global MVCC timestamp</td>
</tr>
<tr>
<td>SECONDARY_SITE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the global MVCC timestamp updated time of the secondary site</td>
</tr>
<tr>
<td>PRIMARY_SITE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the global MVCC updated time of the primary site</td>
</tr>
<tr>
<td>SECONDARY_SITE_UPDATE_DURATION</td>
<td>BIGINT</td>
<td>Displays the global MVCC update duration of the secondary site in milliseconds</td>
</tr>
</tbody>
</table>

Related Information

M_SYSTEM_REPLICATION System View [page 2194]
M_MVCC_OVERVIEW System View [page 2027]
M_MVCC_SNAPSHOTS System View [page 2029]
M_MVCC_TABLES System View [page 2030]
ALTER SYSTEM (ENABLE | DISABLE) SYSTEM REPLICATION Statement (System Management) [page 537]
### 6.2.311 M_SYSTEM_REPLICATION_TAKEOVER_HISTORY

#### System View

Provides access to a history of HSR takeover executions.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAKEOVER_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time of the takeover command. This value matches tenant takeovers that are executed within the same system takeover process.</td>
</tr>
<tr>
<td>TAKEOVER_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the end time of the takeover command.</td>
</tr>
<tr>
<td>EXECUTION_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the execution start time for takeover of the transaction domain.</td>
</tr>
<tr>
<td>EXECUTION_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the execution end time for takeover of the transaction domain.</td>
</tr>
<tr>
<td>SITE_ID</td>
<td>INTEGER</td>
<td>Displays the generated ID of the secondary site at takeover time.</td>
</tr>
<tr>
<td>SITE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the logical name provided by the site administrator at takeover time.</td>
</tr>
<tr>
<td>MASTER_NAMESERVER_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the master nameserver host at takeover time.</td>
</tr>
<tr>
<td>VERSION</td>
<td>VARCHAR(32)</td>
<td>Displays the SAP HANA version for the site that is executing the takeover.</td>
</tr>
<tr>
<td>SOURCE_SITE_ID</td>
<td>INTEGER</td>
<td>Displays the generated ID of the source site at takeover time.</td>
</tr>
<tr>
<td>SOURCE_SITE_NAME</td>
<td>VARCHAR(64)</td>
<td>Displays the logical name for the source site provided by the site administrator at takeover time.</td>
</tr>
<tr>
<td>SOURCE_MASTER_NAMESERVER_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the source site master nameserver host at takeover time.</td>
</tr>
<tr>
<td>SOURCE_VERSION</td>
<td>VARCHAR(32)</td>
<td>Displays the source site SAP HANA version.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TAKEOVER_TYPE</td>
<td>VARCHAR(10)</td>
<td>Displays how the system went online:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ONLINE: an online takeover</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• OFFLINE: an offline takeover</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TIMETRAVEL: a timetravel takeover</td>
</tr>
<tr>
<td>REPLICATION_MODE</td>
<td>VARCHAR(16)</td>
<td>Displays the replication mode at takeover time.</td>
</tr>
<tr>
<td>OPERATION_MODE</td>
<td>VARCHAR(32)</td>
<td>Displays the operation mode at takeover time.</td>
</tr>
<tr>
<td>REPLICATION_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the replication status at takeover time.</td>
</tr>
<tr>
<td>LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the master log position, that has been reached by takeover.</td>
</tr>
<tr>
<td>LOG_POSITION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time reached by the takeover.</td>
</tr>
<tr>
<td>SHIPPED_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the highest master log position that has been shipped before executing takeover.</td>
</tr>
<tr>
<td>SHIPPED_LOG_POSITION_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last shipped log buffer before executing takeover.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays a comment for the remote subscription.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_SYSTEM_OVERVIEW System View [page 2193]
- M_SYSTEM_REPLICATION System View [page 2194]
6.2.312 M_SYSTEM_REPLICATION_TIMETRAVEL System View

Provides information about the valid time travel range for each service on a secondary site.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>START_SNAPSHOT_ID</td>
<td>INTEGER</td>
<td>Displays the snapshot ID of the oldest commit timestamp on which timetravel can start.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the oldest commit timestamp on which timetravel can start.</td>
</tr>
<tr>
<td>START_REDO_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the oldest log position on which timetravel can start.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the newest commit timestamp which can be reached via timetravel.</td>
</tr>
<tr>
<td>END_REDO_LOG_POSITION</td>
<td>BIGINT</td>
<td>Displays the newest log position that can be reached via timetravel.</td>
</tr>
<tr>
<td>COORDINATOR_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the coordinator type: MASTER if it is a transaction master volume type or NONE if it is not a coordinator volume type.</td>
</tr>
</tbody>
</table>

Related Information

M_SYSTEM_OVERVIEW System View [page 2193]
M_SYSTEM_REPLICATION System View [page 2194]
Application-Time Period Tables
6.2.313 M_TABLES System View

Provides information on row and column tables, as well as collection tables (JSON Document Store).

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of records in this table.</td>
</tr>
<tr>
<td>TABLE_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated memory size for fixed-size and variable-size part in bytes.</td>
</tr>
<tr>
<td>IS_COLUMN_TABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is a column table: TRUE/FALSE.</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of the table: ROW, COLUMN, or COLLECTION.</td>
</tr>
<tr>
<td>IS_PARTITIONED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is partitioned: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_REPLICATED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is replicated: TRUE/FALSE.</td>
</tr>
<tr>
<td>HAS_RECORD_COMMIT_TIMESTAMP</td>
<td>VARCHAR(5)</td>
<td>Displays whether table is tracking commit timestamps: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Related Information

TABLES System View [page 1669]
CREATE TABLE Statement (Data Definition) [page 825]
LOCK TABLE Statement (Transaction Management) [page 1056]
RENAME TABLE Statement (Data Definition) [page 1094]
ALTER TABLE Statement (Data Definition) [page 589]
TRUNCATE TABLE Statement (Data Manipulation) [page 1178]
DROP TABLE Statement (Data Definition) [page 980]
Managing Tables
6.2.314  M_TABLE_LOB_FILES System View

Provides information about all LOB files that belong to a table.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the partition ID. Returns the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ble and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is in progress.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the table OID, same as owner_oid if the table is found; 0 other-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>wise.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID.</td>
</tr>
<tr>
<td>PHYSICAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the file size in bytes.</td>
</tr>
<tr>
<td>BINARY_SIZE</td>
<td>BIGINT</td>
<td>Displays the LOB size in bytes.</td>
</tr>
<tr>
<td>CHARACTER_SIZE</td>
<td>INTEGER</td>
<td>Displays the number of characters in NCLOB.</td>
</tr>
<tr>
<td>PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of pages used for a LOB file.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>LOB_TYPE</td>
<td>VARCHAR(5)</td>
<td>Displays the type of LOB.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_TABLE_LOB_STATISTICS System View [page 2202]
- Large Object (LOB) Data Types [page 45]
- ALTER TABLE Statement (Data Definition) [page 589]
- Hybrid LOBs (Large Objects)

**6.2.315 M_TABLE_LOB_STATISTICS System View**

Provides information about the aggregated file and packed LOB statistics per host, port, table, partition, and column.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the name of the host.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name associated with the LOBs.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name associated with the LOBs.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Returns the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For replicated tables, the part ID is 1 for the original table and sub-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sequence part IDs are assigned to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is currently in progress.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name associated with the LOBs.</td>
</tr>
<tr>
<td>LOB_STORAGE_TYPE</td>
<td>VARCHAR(7)</td>
<td>Displays that the LOB storage type is one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>INPLACE</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Small LOBs (completely loaded into the main memory).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PACKED</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Medium-sized LOBs where the size is between two thresholds (only loaded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>into the main memory when required).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>FILE</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Large LOBs (individually loaded into the main memory on demand).</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID of the packedLOB container; NULL for file LOBs.</td>
</tr>
<tr>
<td>DISK_SIZE</td>
<td>BIGINT</td>
<td>Displays the number of bytes used for the LOBs on a disk.</td>
</tr>
<tr>
<td>BINARY_SIZE</td>
<td>BIGINT</td>
<td>Displays the binary size in bytes of the LOBs. This value may be less than</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the DISK_SIZE due to management information and paging.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CHARACTER_SIZE</td>
<td>BIGINT</td>
<td>Displays the number of characters in the NCLOBs (only) column. For packed LOBs, this corresponds to the total number of characters of all of the LOBs in the container.</td>
</tr>
<tr>
<td>LOB_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of associated LOBs.</td>
</tr>
<tr>
<td>READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the counter for read operations on the LOBs.</td>
</tr>
<tr>
<td>MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the size in bytes of the LOB pages that are loaded in the memory.</td>
</tr>
<tr>
<td>MEMORY_PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of LOB pages loaded in the memory.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_TABLE_LOB_FILES System View [page 2201]
- Large Object (LOB) Data Types [page 45]

### 6.2.316 M_TABLE_LOCATIONS System View

Provides information about tables and their logical location. Physical locations are shown in M_TABLE_PERSISTENCE_LOCATIONS.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where table data is located. This value is empty for views.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the port where the table data is located. This value is 0 for views.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Returns the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For replicated tables, the part ID is 1 for the original table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema is currently in progress.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>VARCHAR(75)</td>
<td>Displays the host and port where the table data is located.</td>
</tr>
</tbody>
</table>

**Related Information**

- ALTER SYSTEM ALTER TABLE PLACEMENT Statement (System Management) [page 508]
- CREATE TABLE Statement (Data Definition) [page 825]
- ALTER TABLE Statement (Data Definition) [page 589]
- Set an Explicit Table Location
- M_TABLE_PERSISTENCE_LOCATIONS System View [page 2212]
- TABLE_PLACEMENT System View [page 1684]
- M_TABLE_PLACEMENT_LOCATIONS System View [page 2217]
- M_TABLE_LOCATIONS System View [page 2204]
- TABLE_PLACEMENT_LOCATIONS System View [page 1685]
6.2.317 M_TABLE_LOCKS System View

Shows table locks.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>LOCK_OWNER_TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction object ID owning the lock.</td>
</tr>
<tr>
<td>LOCK_OWNER_UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the write transaction ID owning the lock.</td>
</tr>
<tr>
<td>ACQUIRED_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the lock acquisition time.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the object type.</td>
</tr>
<tr>
<td>LOCK_MODE</td>
<td>VARCHAR(32)</td>
<td>Displays the lock mode: EXCLUSIVE/INTENTIONAL EXCLUSIVE.</td>
</tr>
</tbody>
</table>

Related Information

- LOCK TABLE Statement (Transaction Management) [page 1056]
- M_READWRITELOCKS System View [page 2060]
6.2.318 M_TABLE_PARTITION_STATISTICS System View

Returns the table partition runtime statistics for column store partition tables only. This view is empty if the partition_statistics_select_enabled property in the indexserver.ini configuration file is disabled.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ble and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that the table schema is currently in be-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ing modified.</td>
</tr>
<tr>
<td>SELECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times a partition has been selected.</td>
</tr>
<tr>
<td>LAST_SELECT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last time that this partition was selected.</td>
</tr>
</tbody>
</table>

Related Information

M_TABLE_PARTITIONS System View [page 2209]
M_EXPENSIVE_STATEMENTS System View [page 1905]
Table Partitioning
TABLE_PARTITIONS System View [page 1681]
CREATE TABLE Statement (Data Definition) [page 825]
### 6.2.318.1 M_TABLE_PARTITION_OPERATIONS System View

Provides information on table partition operations from the memory ring buffer or disk trace files.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>START_TIME</td>
<td>LONGDATE</td>
<td>Displays the time the statement started.</td>
</tr>
<tr>
<td>DURATION</td>
<td>BIGINT</td>
<td>Displays the duration, in microseconds, the statement ran.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name who executed the operation.</td>
</tr>
<tr>
<td>STATEMENT_HASH</td>
<td>VARCHAR(32)</td>
<td>Displays the unique identifier for a statement string.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Displays the statement string.</td>
</tr>
<tr>
<td>PARTITION_DEFINITION</td>
<td>NCLOB</td>
<td>Displays the partition definition after executing the statement.</td>
</tr>
<tr>
<td>OPERATION_TYPE</td>
<td>VARCHAR(64)</td>
<td>Displays the operation type strings. Valid strings include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ADD PARTITION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DROP PARTITION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ALTER TABLE PARTITION BY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MOVE PARTITION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ENABLE DYNAMIC RANGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DISABLE DYNAMIC RANGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ALTER PARTITION PROPERTY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MERGE PARTITIONS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ADD PARTITION FROM OTHERS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ADD PARTITION FROM OTHERS FOR INTERVAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DROP EMPTY PARTITIONS</td>
</tr>
<tr>
<td>IS_ONLINE</td>
<td>VARCHAR(5)</td>
<td>Indicates whether DDL is online (TRUE) or offline (FALSE).</td>
</tr>
<tr>
<td>APPLICATION_SOURCE</td>
<td>NVARCHAR(256)</td>
<td>Displays source information of the application.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user name.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(45)</td>
<td>Displays the IP address of the client machine.</td>
</tr>
</tbody>
</table>
Additional Information

Users with the CATALOG READ system privilege can view information on all records in this system view. Users without this system privilege can only view information on tables that they own.

6.2.319 M_TABLE_PARTITIONS System View

Provides information regarding partition-specific memory and disk usage for partitioned tables.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>NODE_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the partition node.</td>
</tr>
<tr>
<td>PARENT_NODE_ID</td>
<td>INTEGER</td>
<td>Displays the ID of the parent node.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For replicated tables, the part ID is 1 for the original table and sub-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is currently in progress.</td>
</tr>
<tr>
<td>LOAD_UNIT</td>
<td>VARCHAR(7)</td>
<td>Displays the effective load unit. Valid values are PAGE (when the partition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>uses page loadable storage) and COLUMN (when the partition uses column</td>
</tr>
<tr>
<td></td>
<td></td>
<td>loadable storage).</td>
</tr>
<tr>
<td>STORAGE_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays DEFAULT when a column storage type is used. Displays EXTENDED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>when an extended storage type is used.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>Displays the total memory size in bytes, which is the sum of the memory in the main, delta, and history parts.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_MAIN</td>
<td>BIGINT</td>
<td>Displays the current memory consumption in main in bytes. This value varies depending on the number of loaded attributes and includes data for open transactions.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_PAGE_LOADABLE</td>
<td>BIGINT</td>
<td>Displays the total resident paged memory size, in bytes, of the partition.</td>
</tr>
<tr>
<td>MEMORY_SIZE_IN_DELTA</td>
<td>BIGINT</td>
<td>Displays the current memory consumption in delta in bytes.</td>
</tr>
<tr>
<td>PERSISTENT_MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>Displays the current pure persistent memory consumption (in bytes) for every column store table</td>
</tr>
<tr>
<td>DISK_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of a table partition on a disk in bytes. Column storage displays the size on the disk per partition. Extended storage displays the size on the disk for all extended storage partitions.</td>
</tr>
<tr>
<td>DISK_SIZE_IN_PAGE_LOADABLE</td>
<td>BIGINT</td>
<td>Displays the total disk size, in bytes, of page-loadable storage for this partition.</td>
</tr>
<tr>
<td>ESTIMATED_MAX_MEMORY_SIZE_IN_TOTAL</td>
<td>BIGINT</td>
<td>Displays the total estimated maximum memory consumption in bytes for fully loaded partitions. This value does not include data for open transactions.</td>
</tr>
<tr>
<td>LAST_ESTIMATED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the last estimated memory consumption in bytes of a fully loaded partition.</td>
</tr>
<tr>
<td>LAST_ESTIMATED_MEMORY_SIZE_TIME</td>
<td>TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the record count.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RAW_RECORD_COUNT_IN_MAIN</td>
<td>BIGINT</td>
<td>Displays the current number of entries in the main part. This value differs from the number of visible main rows because there are entries for modified rows that are marked as invalid. Displays the last time that the last-estimated memory consumption was calculated.</td>
</tr>
<tr>
<td>RAW_RECORD_COUNT_IN_DELTA</td>
<td>BIGINT</td>
<td>Displays the current number of entries in the delta part. This value differs from the number of visible delta rows because there are additional entries, such as deleted rows or updated rows. This value can contain deleted records.</td>
</tr>
<tr>
<td>LOADED</td>
<td>VARCHAR(10)</td>
<td>Displays the last time that the last-estimated memory consumption was calculated. Displays how many columns in the table are loaded in memory. Valuse are: NO, PARTIALLY, or FULL. For a description of each column, see M_CS_COLUMNS.</td>
</tr>
<tr>
<td>IS_REPLICA</td>
<td>VARCHAR(5)</td>
<td>Displays whether the part is a replica (TRUE/FALSE).</td>
</tr>
<tr>
<td>GROUP_TYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the effective group type.</td>
</tr>
<tr>
<td>SUBTYPE</td>
<td>NVARCHAR(256)</td>
<td>Displays the effective group subtype.</td>
</tr>
<tr>
<td>GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the effective group name.</td>
</tr>
</tbody>
</table>

Related Information

Table Partitioning
TABLE PARTITIONS System View [page 1681]
M_TABLE_PARTITION_STATISTICS System View [page 2207]
M_CS_COLUMNS System View [page 1814]
CREATE TABLE Statement (Data Definition) [page 825]
6.2.320 M_TABLE_PERSISTENCE_LOCATIONS System View

Provides information about column store tables and their physical data locations.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ble and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that the table schema is currently in be-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ing modified.</td>
</tr>
<tr>
<td>IS_HISTORY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the history is part of a table: TRUE/FALSE.</td>
</tr>
<tr>
<td>PERSISTENCE_HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host where table data is located.</td>
</tr>
<tr>
<td>PERSISTENCE_PORT</td>
<td>INTEGER</td>
<td>Displays the port where table data is located.</td>
</tr>
</tbody>
</table>

Additional Information

This view shows information on which node contains the persistence parts of a table. This includes the assigned node as visible from the M_TABLE_LOCATIONS view, but also includes other nodes that still contain some persistence of the table. This could occur if a table is moved and not yet merged. In this case the old node still contains some persistence content for the table beside the currently assigned node.
## 6.2.321 M_TABLE_PERSISTENCE_LOCATION_STATISTICS System View

Provides persistence storage statistics for tables partitions and services.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that the table schema is currently being</td>
</tr>
<tr>
<td></td>
<td></td>
<td>modified.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>DISK_SIZE</td>
<td>BIGINT</td>
<td>Displays the total disk size, in bytes, of all table parts.</td>
</tr>
<tr>
<td>PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of pages of all table parts.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BYTES_WRITTEN</td>
<td>BIGINT</td>
<td>Displays the number of bytes written to the table.</td>
</tr>
<tr>
<td>BYTES_APPENDED</td>
<td>BIGINT</td>
<td>Displays the number of bytes appended to the table.</td>
</tr>
<tr>
<td>BYTES_READ</td>
<td>BIGINT</td>
<td>Displays the number of bytes read from the table.</td>
</tr>
<tr>
<td>BYTESTREAM_WRITTEN</td>
<td>BIGINT</td>
<td>Displays the number of bytes written to the table via the streaming interface.</td>
</tr>
<tr>
<td>APPEND_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times the table was appended to.</td>
</tr>
<tr>
<td>WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times the table was written to.</td>
</tr>
<tr>
<td>OPTIMIZE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times the table was written to be optimized.</td>
</tr>
<tr>
<td>READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times the table was read from.</td>
</tr>
<tr>
<td>TRUNCATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times the table was truncated.</td>
</tr>
<tr>
<td>COPY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times the table was copied.</td>
</tr>
</tbody>
</table>

### Related Information

- **M_TABLE_PERSISTENCE_STATISTICS System View** [page 2215]
- **M_EXPENSIVE_STATEMENT_EXECUTION_LOCATION_STATISTICS System View** [page 1910]
- Persistent Memory
- Persistent Data Storage in the SAP HANA Database
6.2.322  M_TABLE_PERSISTENCE_STATISTICS System View

Provides persistence summary statistics for tables.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>DISK_SIZE</td>
<td>BIGINT</td>
<td>Displays the total disk size, in bytes, of all of the table parts.</td>
</tr>
<tr>
<td>PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of pages of all of the table parts.</td>
</tr>
<tr>
<td>BYTES_WRITTEN</td>
<td>BIGINT</td>
<td>Displays the number of bytes that are written to the table.</td>
</tr>
<tr>
<td>BYTES_APPENDED</td>
<td>BIGINT</td>
<td>Displays the number of bytes that are appended to the table.</td>
</tr>
<tr>
<td>BYTES_READ</td>
<td>BIGINT</td>
<td>Displays the number of bytes that are read from the table.</td>
</tr>
<tr>
<td>BYTESTREAM_WRITTEN</td>
<td>BIGINT</td>
<td>Displays the number of bytes that are written to the table via a streaming interface.</td>
</tr>
<tr>
<td>APPEND_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times that the table was appended to.</td>
</tr>
<tr>
<td>WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times that the table was written to.</td>
</tr>
<tr>
<td>OPTIMIZE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times that the table was optimized.</td>
</tr>
<tr>
<td>READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times that the table was read from.</td>
</tr>
<tr>
<td>TRUNCATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times that the table was truncated.</td>
</tr>
<tr>
<td>COPY_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of times that the table was copied.</td>
</tr>
</tbody>
</table>
Related Information

CREATE STATISTICS Statement (Data Definition) [page 815]
ALTER STATISTICS Statement (Data Definition) [page 484]
REFRESH STATISTICS Statement (Data Definition) [page 1082]
DROP STATISTICS Statement (Data Definition) [page 974]
Persistent Memory

6.2.323 M_TABLE_PERSISTENT_MEMORY_FILES System View

Displays the persistent memory file information for all of the tables in the database.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Returns the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For replicated tables, the part ID is 1 for the original table and sub-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>quent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that a modification of the table schema</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is currently in progress.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>COLUMN_FRAGMENT_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the column fragment type: PAGED MAIN, SINGLE MAIN, or DELTA.</td>
</tr>
</tbody>
</table>
### Related Information

<table>
<thead>
<tr>
<th>ALTER TABLE Statement (Data Definition) [page 589]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE TABLE Statement (Data Definition) [page 825]</td>
</tr>
<tr>
<td>EXPORT Statement (Data Import Export) [page 997]</td>
</tr>
<tr>
<td>Non-heterogeneous Alter Partition Clauses [page 630]</td>
</tr>
<tr>
<td>Non-heterogeneous Create Partition Clauses [page 863]</td>
</tr>
<tr>
<td>UNLOAD Statement (Data Manipulation) [page 1179]</td>
</tr>
<tr>
<td>CREATE TABLE Statement (Multistore Table) [Dynamic Tiering] [page 1379]</td>
</tr>
<tr>
<td>IMPORT Statement (Data Import Export) [page 1029]</td>
</tr>
</tbody>
</table>

### 6.2.324 M_TABLE_PLACEMENT_LOCATIONS System View

Provides table placement location monitoring information.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the location.</td>
</tr>
<tr>
<td>SYSTEM_DEFINED_VOLUME_IDS</td>
<td>NVARCHAR(256)</td>
<td>Displays the volume IDs, which are received from the topology.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INCLUDE</td>
<td>NVARCHAR(256)</td>
<td>Displays the list of volume IDs or synonyms which are added to the location name.</td>
</tr>
<tr>
<td>EXCLUDE</td>
<td>NVARCHAR(256)</td>
<td>Displays the list of volume IDs or synonyms which are removed from the location name.</td>
</tr>
<tr>
<td>EFFECTIVE_Volume_IDS</td>
<td>NVARCHAR(256)</td>
<td>Displays the calculated volume IDs for the location name.</td>
</tr>
</tbody>
</table>

Related Information

ALTER SYSTEM ALTER TABLE PLACEMENT Statement (System Management) [page 508]
CREATE TABLE Statement (Data Definition) [page 825]
ALTER TABLE Statement (Data Definition) [page 589]
Table Placement Rules for Replicas
TABLE_PLACEMENT System View [page 1684]
M_TABLE_LOCATIONS System View [page 2204]
M_TABLE_PLACEMENT_LOCATIONS System View [page 2217]

6.2.325 M_TABLE_PRUNING_STATISTICS System View

Provides an interface to access statistics for historical data. This data helps detect unexpectedly heavy historical processing by an application, including disclosure of unqualified historical queries as well as assessing the effectiveness of pruning.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the statement ID.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PRUNING_CALLER</td>
<td>VARCHAR(20)</td>
<td>Displays the operation that initiated the pruning: SEARCH, OLAP ENGINE, JOIN ENGINE, TABLE UPDATE, SQL PLAN GENERATION, or SQL PLAN EXECUTION.</td>
</tr>
<tr>
<td>PRUNING_TYPE</td>
<td>VARCHAR(20)</td>
<td>Displays the type of pruning that was initiated: COMPILE TIME, RUNTIME STATIC, RANGE RESTRICTION, or RUNTIME DYNAMIC.</td>
</tr>
<tr>
<td>PRUNING_FILTER</td>
<td>NCLOB</td>
<td>Displays the predicate used for pruning.</td>
</tr>
<tr>
<td>TOTAL_CS_PART_COUNT</td>
<td>INTEGER</td>
<td>Displays the total number of column store partitions before pruning.</td>
</tr>
<tr>
<td>TOTAL_ES_PART_COUNT</td>
<td>INTEGER</td>
<td>Displays the total number of extended storage partitions before pruning.</td>
</tr>
<tr>
<td>PRUNED_CS_PART_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of column store partitions after pruning.</td>
</tr>
<tr>
<td>PRUNED_CS_PAGE_LOADABLE_PART_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of column store partitions using paged attributes after pruning.</td>
</tr>
<tr>
<td>PRUNED_ES_PART_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of extended storage partitions after pruning.</td>
</tr>
</tbody>
</table>

**Related Information**

Static and Dynamic Partition Pruning  
Table Partitioning
## 6.2.326 M_TABLE_REPLICAS System View

Provides detailed information on asynchronous/synchronous table replicas.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the replica (replication target).</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of the replica (replication target).</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>VARCHAR(6)</td>
<td>Displays the replica table type: ROW/COLUMN.</td>
</tr>
<tr>
<td>REPLICA_TYPE</td>
<td>VARCHAR(12)</td>
<td>Displays the replication type: ASYNCHRONOUS/SYNCHRONOUS.</td>
</tr>
<tr>
<td>SOURCE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the replication source.</td>
</tr>
<tr>
<td>SOURCE_TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of the replication source.</td>
</tr>
<tr>
<td>SOURCE_TABLE_TYPE</td>
<td>VARCHAR(6)</td>
<td>Displays the source table type: ROW/COLUMN.</td>
</tr>
<tr>
<td>SOURCE_TABLE_VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID of the source table.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original ta-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ble and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that the table schema is currently in be-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ing modified.</td>
</tr>
<tr>
<td>REPLICATION_STATUS</td>
<td>VARCHAR(8)</td>
<td>Displays the table replication status: ENABLED, ENABLING, or DISABLED.</td>
</tr>
<tr>
<td>LAST_ENABLE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last replication enablement.</td>
</tr>
<tr>
<td>LAST_DISABLE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last replication disablement.</td>
</tr>
<tr>
<td>LAST_ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the last error code.</td>
</tr>
<tr>
<td>LAST_ERROR_MESSAGE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the last error message.</td>
</tr>
<tr>
<td>LAST_ERROR_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last error.</td>
</tr>
<tr>
<td>INSERT_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the inserted record count.</td>
</tr>
<tr>
<td>UPDATE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the updated record count.</td>
</tr>
<tr>
<td>DELETE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Displays the deleted record count.</td>
</tr>
<tr>
<td>INSERT_RETRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the insert statement retry count.</td>
</tr>
<tr>
<td>UPDATE_RETRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the update statement retry count.</td>
</tr>
<tr>
<td>DELETE_RETRY_COUNT</td>
<td>BIGINT</td>
<td>Displays the delete statement retry count.</td>
</tr>
</tbody>
</table>

**Related Information**

**M_TABLE_REPLICAS_RESET System View** [page 2222]
### 6.2.327 M_TABLE_REPLICAS_RESET System View

Provides detailed information on asynchronous/synchronous table replicas.

This view contains values accumulated since the last reset of the main view M_TABLE_REPLICAS. Refer to M_TABLE_REPLICAS for information about the structure and use of this view.

In addition to the members mentioned in M_TABLE_REPLICAS, this view also contains a timestamp field, `RESET_TIME`, which indicates the last time the data was reset.

#### Related Information

M_TABLE_REPLICAS System View [page 2220]

### 6.2.328 M_TABLE_REPLICATION_VOLUME_HISTORY

Displays detailed information on the history of the table replication volume.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence Volume ID.</td>
</tr>
<tr>
<td>SOURCE_VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID of the source. If value is greater than 2048, it means it is in a remote location. This value is applicable to remote table replication only.</td>
</tr>
<tr>
<td>REPLICA_VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID of replica. If the value is greater than 2048, it means that it is a remote location.</td>
</tr>
<tr>
<td>SOURCE_REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name of the source system. This value only appears if SOURCE_VOLUME_ID or REPLICA_VOLUME_ID is greater than 2048.</td>
</tr>
</tbody>
</table>
### 6.2.329 M_TABLE_REPLICATION_VOLUME_STATISTICS

Provides detailed information on table replication volume statistics.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPLICA_REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name of the replica system. This value only appears shown if SOURCE_VOLUME_ID or REPLICA_VOLUME_ID is greater than 2048.</td>
</tr>
<tr>
<td>REPLICATION_STATUS_CHANGE_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time when the replication status was changed.</td>
</tr>
<tr>
<td>REPLICATION_STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the replication status of the replica volume. Values are: DISABLED, ENABLING, ENABLED, ENABLED_BUT_UNAVAILABLE.</td>
</tr>
<tr>
<td>REPLICATION_STATUS_CHANGE_REASON</td>
<td>VARCHAR(32)</td>
<td>Displays the reason for the change in replication status. Values are: EXCEPTION, EXCEPTION_CLEANUP, COMMAND, RESTART, SHUTDOWN, INTERNAL, INITIAL.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the error code in the event a replication volume related to the exception is turned off.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the error message in case the event a replication volume related to the exception is turned off.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence Volume ID.</td>
</tr>
<tr>
<td>SOURCE_REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name of source system. If SOURCE_REMOTE_SOURCE_NAME and REPLICA_REMOTE_SOURCE_NAME are NULL, it means non-remote table replication (RTR).</td>
</tr>
<tr>
<td>REPLICA_REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name of replica system.</td>
</tr>
<tr>
<td>LAST_SOURCE_COMMIT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last commit time of source system. The value in this column cannot be reset using the ALTER SYSTEM RESET MONITORING View statement.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TRANSACTION_LOG_QUEUE_SIZE</td>
<td>BIGINT</td>
<td>Displays the current transaction log queue size. This is the log count in the queue. The value in this column cannot be reset using the ALTER SYSTEM RESET MONITORING View statement.</td>
</tr>
<tr>
<td>SHIPPED_TRANSACTION_LOG_COUNT</td>
<td>BIGINT</td>
<td>Displays the shipped transaction log count.</td>
</tr>
<tr>
<td>SHIPPED_TRANSACTION_LOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the shipped transaction log size in bytes.</td>
</tr>
<tr>
<td>REPLAYED_TRANSACTION_LOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the replayed transaction log size in bytes.</td>
</tr>
<tr>
<td>REPLAYED_PRECOMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the replayed precommit count.</td>
</tr>
<tr>
<td>REPLAYED_COMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the replayed commit count. The count includes the replayed DDL/DML commit count.</td>
</tr>
<tr>
<td>REPLAYED_ROLLBACK_COUNT</td>
<td>BIGINT</td>
<td>Displays the replayed rollback count.</td>
</tr>
<tr>
<td>REPLAYED_DDL_COMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the replayed DDL commit count. This is applicable to remote table replication only.</td>
</tr>
<tr>
<td>TRANSACTION_REPLAY_ERROR_COUNT</td>
<td>BIGINT</td>
<td>Displays the transaction replay error count.</td>
</tr>
<tr>
<td>AVG_PRECOMMIT_PROPAGATION_LATENCY</td>
<td>BIGINT</td>
<td>Displays the average precommit propagation latency time, in microseconds.</td>
</tr>
<tr>
<td>AVG_PROPAGATION_LATENCY</td>
<td>BIGINT</td>
<td>Displays the average commit propagation latency, in microseconds. Propagation latency is the sum of network latency, dispatch latency, and processing latency.</td>
</tr>
<tr>
<td>AVG_PROCESSING_LATENCY</td>
<td>BIGINT</td>
<td>Displays the average processing latency, in microseconds, of the commit log.</td>
</tr>
<tr>
<td>TOTAL_DML_WAIT_TIME</td>
<td>BIGINT</td>
<td>Displays the accumulated DML wait time, in microseconds.</td>
</tr>
</tbody>
</table>

### 6.2.330 M_TABLE_REPLICATION_VOLUME_STATISTICS_RESET

Displays detailed information on values after executing ALTER SYSTEM RESET MONITORING VIEW.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the time of the last reset.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence Volume ID.</td>
</tr>
<tr>
<td>SOURCE_REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name of source system. If SOURCE_REMOTE_SOURCE_NAME and REPLICA_REMOTE_SOURCE_NAME are NULL, it means non-remote table replication (RTR).</td>
</tr>
<tr>
<td>REPLICA_REMOTE_SOURCE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the remote source name of replica system.</td>
</tr>
<tr>
<td>LAST_SOURCE_COMMIT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the last commit time of source system. The value in this column cannot be reset using the ALTER SYSTEM RESET MONITORING View statement.</td>
</tr>
<tr>
<td>TRANSACTION_LOG_QUEUE_SIZE</td>
<td>BIGINT</td>
<td>Displays the current transaction log queue size. This is the log count in the queue. The value in this column cannot be reset using the ALTER SYSTEM RESET MONITORING View statement.</td>
</tr>
<tr>
<td>SHIPPED_TRANSACTION_LOG_COUNT</td>
<td>BIGINT</td>
<td>Displays the shipped transaction log count.</td>
</tr>
<tr>
<td>SHIPPED_TRANSACTION_LOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the shipped transaction log size in bytes.</td>
</tr>
<tr>
<td>REPLAYED_TRANSACTION_LOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the replayed transaction log size in bytes.</td>
</tr>
<tr>
<td>REPLAYED_PRECOMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the replayed precommit count.</td>
</tr>
<tr>
<td>REPLAYED_COMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the replayed commit count. The count includes the replayed DDL/DML commit count.</td>
</tr>
<tr>
<td>REPLAYED.Rollback_COUNT</td>
<td>BIGINT</td>
<td>Displays the replayed rollback count.</td>
</tr>
<tr>
<td>REPLAYED.DDL_COMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the replayed DDL commit count. This is applicable to remote table replication only.</td>
</tr>
<tr>
<td>TRANSACTION_REPLAY_ERROR_COUNT</td>
<td>BIGINT</td>
<td>Displays the transaction replay error count.</td>
</tr>
<tr>
<td>AVG_PRECOMMIT_PROPAGATION_LATENCY</td>
<td>BIGINT</td>
<td>Displays the average precommit propagation latency, in microseconds.</td>
</tr>
<tr>
<td>AVG_PROPAGATION_LATENCY</td>
<td>BIGINT</td>
<td>Displays the average commit propagation latency, in microseconds. Propagation latency is the sum of network latency, dispatch latency, and processing latency.</td>
</tr>
</tbody>
</table>
### 6.2.331 M_TABLE_SNAPSHOTS System View

Provides snapshot information for tables that are blocked by table-wise GC.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the table object ID.</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the table type: ROW/COLUMN.</td>
</tr>
<tr>
<td>MIN_MVCC_SNAPSHOT_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the minimum MVCC timestamp which is held by the table-wise GC blocker transaction or cursor.</td>
</tr>
<tr>
<td>VERSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of versions (for a column table, this value is the number of unconsolidated udivlists).</td>
</tr>
</tbody>
</table>

#### Related Information

- M_SNAPSHOTS System View [page 2134]
- M_MVCC_SNAPSHOTS System View [page 2029]
6.2.332 M_TABLE_STATISTICS System View

Returns the table runtime statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name to which the table belongs.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name to be counted.</td>
</tr>
<tr>
<td>INSERT_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of INSERT statements for the table.</td>
</tr>
<tr>
<td>DELETE_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of DELETE statements for the table.</td>
</tr>
<tr>
<td>UPDATE_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of UPDATE statements for the table.</td>
</tr>
<tr>
<td>REPLACE_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of REPLACE statements for the table.</td>
</tr>
<tr>
<td>MERGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of MERGE INTO statements for the table.</td>
</tr>
<tr>
<td>SELECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the count of SELECT statements for the table.</td>
</tr>
<tr>
<td>LAST_MODIFY_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the last INSERT, UPDATE, DELETE, or REPLACE statement was executed.</td>
</tr>
<tr>
<td>LAST_SELECT_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the timestamp when the last SELECT statement was executed.</td>
</tr>
</tbody>
</table>

Additional Information

- The M_TABLE_STATISTICS system view is not transactional; when a DML statement is rolled back, count values in the view remain unchanged.
- DML statements that do not result in changes to data still increment the DML counters for the relevant tables.
- DML-related counters and SELECT_COUNT are only applicable to user tables.
• Generally, when a single SELECT statement touches more than one table (regardless of whether results are returned from those tables) the SELECT_COUNTER for each table touched is incremented. However, SELECT statements that are subqueries do not increment the SELECT_COUNT counter for the tables referenced in the subquery. For example, a SELECT statement that is a subquery of an INSERT statement does not increment the SELECT_COUNT counter.

• Nested SELECT statements, for example in procedures, increment the SELECT counter, with one exception: for a SELECT statement in a calculation view, nested SELECT statements cannot increment the SELECT counter when nested SELECT statements are optimized.

This view has a resettable counterpart; you can see the values since the last reset in the M_TABLE_STATISTICS_RESET system view. To reset the view, execute the following statement: ALTER SYSTEM RESET MONITORING VIEW SYS.M_TABLE_STATISTICS_RESET;

Related Information

M_TABLE_STATISTICS_RESET System View [page 2228]
CREATE STATISTICS Statement (Data Definition) [page 815]
REFRESH STATISTICS Statement (Data Definition) [page 1082]
ALTER STATISTICS Statement (Data Definition) [page 484]
DROP STATISTICS Statement (Data Definition) [page 974]
Create Statistics on a Virtual Table or Linked Database
Alter Statistics on a Virtual Table or Linked Database
Refresh Statistics on a Virtual Table or Linked Database
Drop Statistics on a Virtual Table or Linked Database

6.2.333 M_TABLE_STATISTICS_RESET System View

Returns the table DML runtime statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_TABLE_STATISTICS. Refer to M_TABLE_STATISTICS System View for information about the structure and use of this view.

In addition to the members mentioned in the M_TABLE_STATISTICS System View topic, this system view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_TABLE_STATISTICS System View [page 2227]
### M_TABLE_VIRTUAL_FILES System View

Provides information about all virtual files that belong to a table.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Displays the table partition ID:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For partitioned tables, the part ID is equal to the sequential number of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the partition, starting at 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• In the case of replicated tables, the part ID is 1 for the original</td>
</tr>
<tr>
<td></td>
<td></td>
<td>table and subsequent part IDs are assigned to replica tables.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The part ID is 0 for tables that are not partitioned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A part ID value of -1 indicates that the table schema is currently being</td>
</tr>
<tr>
<td></td>
<td></td>
<td>modified.</td>
</tr>
<tr>
<td>IS_HISTORY</td>
<td>VARCHAR(5)</td>
<td>Displays whether history is part of a table: TRUE/FALSE.</td>
</tr>
<tr>
<td>CONTAINER_ID</td>
<td>BIGINT</td>
<td>Displays the container ID.</td>
</tr>
<tr>
<td>NAMESPACE</td>
<td>VARCHAR(512)</td>
<td>Displays the namespace address in the database storage system.</td>
</tr>
<tr>
<td>NAME</td>
<td>VARCHAR(512)</td>
<td>Displays the name in the database storage system.</td>
</tr>
<tr>
<td>PHYSICAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the storage size used for the file in bytes.</td>
</tr>
<tr>
<td>PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of storage pages used for the file.</td>
</tr>
</tbody>
</table>
Related Information

M_TABLES System View [page 2200]
M_TABLE_LOB_FILES System View [page 2201]

6.2.335 M_TASKS System View

Provides task monitoring information.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name used in the task.</td>
</tr>
<tr>
<td>TASK_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the task.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection identifier.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction identifier used for the task execution.</td>
</tr>
<tr>
<td>TASK_EXECUTION_ID</td>
<td>BIGINT</td>
<td>Displays the unique identifier of the task execution.</td>
</tr>
<tr>
<td>PARENT_TASK_EXECUTION_ID</td>
<td>BIGINT</td>
<td>Displays the parent task identifier.</td>
</tr>
<tr>
<td>IS_ASYNC</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not this is an asynchronous task: TRUE/FALSE.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the input parameters for the task.</td>
</tr>
<tr>
<td>PROCEDURE_PARAMETERS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the input procedure parameters for the task.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start time of the task.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the end time of the task.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DURATION</td>
<td>BIGINT</td>
<td>Displays the execution time of the task in microseconds.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the status of the task: STARTING, RUNNING, FAILED, or COMPLETED.</td>
</tr>
<tr>
<td>CURRENT_OPERATION</td>
<td>NVARCHAR(128)</td>
<td>Displays the operation of the task.</td>
</tr>
<tr>
<td>PROCESSED_RECORDS</td>
<td>BIGINT</td>
<td>Displays the total number of processed records.</td>
</tr>
<tr>
<td>PARTITION_COUNT</td>
<td>INTEGER</td>
<td>Displays the total number of task partitions for a task execution. This value is 1 if there is no task partitioning.</td>
</tr>
<tr>
<td>TOTAL_PROGRESS_PERCENT</td>
<td>DOUBLE</td>
<td>Displays the total task progress percentage.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>APPLICATION_USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the application user name.</td>
</tr>
<tr>
<td>HAS_SIDE_EFFECTS</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not the task produces side effect data: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

- TASKS System View [page 1687]
- TASKPARAMETERS System View [page 1689]
6.2.336 M_TEMPORARY_JOIN_CONDITIONS System View

Provides information about temporary join conditions.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join view name.</td>
</tr>
<tr>
<td>JOIN_CONDITION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join condition name.</td>
</tr>
<tr>
<td>JOIN_ORDER</td>
<td>BIGINT</td>
<td>Displays the join order number.</td>
</tr>
<tr>
<td>TABLE_SCHEMA_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of column1.</td>
</tr>
<tr>
<td>TABLE_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of column1.</td>
</tr>
<tr>
<td>COLUMN_NAME1</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of column1.</td>
</tr>
<tr>
<td>ALIAS_NUMBER1</td>
<td>INTEGER</td>
<td>Displays the alias number of table name1.</td>
</tr>
<tr>
<td>TABLE_SCHEMA_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of column2.</td>
</tr>
<tr>
<td>TABLE_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of column2.</td>
</tr>
<tr>
<td>COLUMN_NAME2</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of column2.</td>
</tr>
<tr>
<td>ALIAS_NUMBER2</td>
<td>INTEGER</td>
<td>Displays the alias number of table name2.</td>
</tr>
<tr>
<td>CONSTRAINTS</td>
<td>NVARCHAR(256)</td>
<td>Displays the join constraint name.</td>
</tr>
<tr>
<td>JOIN_TYPE</td>
<td>VARCHAR(5)</td>
<td>Displays the join type.</td>
</tr>
<tr>
<td>CARDINALITY</td>
<td>VARCHAR(3)</td>
<td>Displays the join cardinality.</td>
</tr>
</tbody>
</table>

Related Information

M_TEMPORARY_JOIN_CONSTRAINTS System View [page 2233]
ALTER SYSTEM CLEAR COLUMN JOIN DATA STATISTICS (System Management) [page 520]
6.2.337  M_TEMPORARY_JOIN_CONSTRAINTS System View

Provides information about temporary join constraints.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view name.</td>
</tr>
<tr>
<td>JOIN_CONSTRAINT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join constraint name.</td>
</tr>
<tr>
<td>CONSTRAINT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the operator.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the constraint owner.</td>
</tr>
<tr>
<td>TABLE_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the column.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name of the column.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the column name.</td>
</tr>
<tr>
<td>OPERATOR</td>
<td>NVARCHAR(256)</td>
<td>Displays the operator.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the value.</td>
</tr>
</tbody>
</table>

6.2.338  M_TEMPORARY_KEY_FIGURES System View

Provides information about temporary key figures.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the join view name.</td>
</tr>
</tbody>
</table>
Column name | Data type | Description
---|---|---
KEY_FIGURE_NAME | NVARCHAR(256) | Displays the key figure name.
DEFAULT_AGGREGATION_TYPE | VARCHAR(32) | Displays the aggregation type.
DESCRIPTION | NVARCHAR(5000) | Displays the description.
UNIT_CONVERSION_NAME | NVARCHAR(256) | Displays the name of the unit conversion.
TABLE_SCHEMA_NAME | NVARCHAR(256) | Displays the schema name of the column.
TABLE_NAME | NVARCHAR(256) | Displays the table name of the column.
COLUMN_NAME | NVARCHAR(256) | Displays the column name.
EXPRESSION_FLAGS | VARCHAR(32) | Displays the expression flags.
EXPRESSION | NVARCHAR(256) | Displays the expression.

Related Information

CS_KEY_FIGURES System View [page 1533]

6.2.339 M_TEMPORARY_OBJECT_DEPENDENCIES System View

Provides information about temporary object dependencies for transient objects.

Structure

Column name | Data type | Description
---|---|---
BASE_DATABASE_NAME | NVARCHAR(256) | Displays the database name of the base object.
BASE_SCHEMA_NAME | NVARCHAR(256) | Displays the schema name of the base object.
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of the base object.</td>
</tr>
<tr>
<td>BASE_OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the type of base object.</td>
</tr>
<tr>
<td>BASE_OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the base object.</td>
</tr>
<tr>
<td>BASE_OBJECT_IS_TEMPORARY</td>
<td>VARCHAR(5)</td>
<td>Displays the temporary property of the base object.</td>
</tr>
<tr>
<td>DEPENDENT_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the database name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_OBJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the object name of the dependent object.</td>
</tr>
<tr>
<td>DEPENDENT_OBJECT_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the dependent type.</td>
</tr>
<tr>
<td>DEPENDENT_OBJECT_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the dependent.</td>
</tr>
<tr>
<td>DEPENDENT_OBJECT_IS_TEMPORARY</td>
<td>VARCHAR(5)</td>
<td>Displays the temporary property of the dependent.</td>
</tr>
</tbody>
</table>

**Related Information**

OBJECTS System View [page 1603]
OBJECT_DEPENDENCIES System View [page 1604]
Object Privileges
Object Dependencies View Examples
6.2.340 M_TEMPORARY_TABLES System View

Provides information about temporary tables.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the comment on the table.</td>
</tr>
<tr>
<td>FIXED_PART_SIZE</td>
<td>SMALLINT</td>
<td>Displays the fixed-size part of the record in bytes.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the anchor global session ID (anchor connection ID).</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Displays the statement ID of the CREATE statement.</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>INTEGER</td>
<td>Displays the record count.</td>
</tr>
<tr>
<td>TABLE_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory consumed by the table in bytes.</td>
</tr>
<tr>
<td>IS_LOGGED</td>
<td>VARCHAR(5)</td>
<td>Displays whether logging is enabled for the table: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_SYSTEM_TABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is a system table: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_COLUMN_TABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is a column table: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_INSERT_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is an insert only table: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_TENANT_SHARED_DATA</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table can be shared among other instances: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_TENANT_SHARED_METADATA</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is a global table: TRUE/FALSE.</td>
</tr>
<tr>
<td>SESSION_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the session type: NONE, SIMPLE, or HISTORY. For HISTORY, time travel is possible.</td>
</tr>
<tr>
<td>IS_TEMPORARY</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table is a temporary table: TRUE/FALSE.</td>
</tr>
<tr>
<td>TEMPORARY_TABLE_TYPE</td>
<td>VARCHAR(8)</td>
<td>Displays the temporary table type.</td>
</tr>
<tr>
<td>IS_USER_DEFINED_TYPE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the user defined a table type: TRUE/FALSE.</td>
</tr>
<tr>
<td>PRELOAD</td>
<td>VARCHAR(5)</td>
<td>Displays whether the table uses pre-loading: TRUE/FALSE. This is only valid for column store tables.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE TABLE Statement (Data Definition) [page 825]
M_TEMPORARY_TABLE_COLUMNS System View [page 2238]
TABLES System View [page 1669]
ALTER TABLE Statement (Data Definition) [page 589]
Managing Tables
Handling Temporary Data
### 6.2.341 M_TEMPORARY_TABLE_COLUMNS System View

Provides information about temporary table columns.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the table name.</td>
</tr>
<tr>
<td>TABLE_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the table.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the column.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the table column.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the SQL data type ID of the column.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the SQL data type name of the column.</td>
</tr>
<tr>
<td>OFFSET</td>
<td>SMALLINT</td>
<td>Displays the offset of the column in the record in bytes.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the number of characters for character types. This value is the number of max digits for numeric types, the number of characters for datetime types, or the number of bytes for LOB types.</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Displays the maximum number of digits to the right of the decimal point: TIME or TIMESTAMP. The decimal digits are defined as the number of digits to the right of the decimal point in the second's component of the data.</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is allowed to accept a null value: TRUE/FALSE.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the default value of the column in bytes.</td>
</tr>
<tr>
<td>COLLATION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the collation of the column.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the comments on the column.</td>
</tr>
<tr>
<td>MAX_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the maximum value in bytes.</td>
</tr>
<tr>
<td>MIN_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the minimum value in bytes.</td>
</tr>
<tr>
<td>CS_DATA_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the column store data type ID.</td>
</tr>
<tr>
<td>CS_DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the column store data type name.</td>
</tr>
<tr>
<td>DDIC_DATA_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the DDIC data type ID.</td>
</tr>
<tr>
<td>DDIC_DATA_TYPE_NAME</td>
<td>VARCHAR(7)</td>
<td>Displays the DDIC data type name.</td>
</tr>
<tr>
<td>COMPRESSION_TYPE</td>
<td>VARCHAR(9)</td>
<td>Displays the type of compression: DEFAULT, PREFIXED, SPARSE, CLUSTERED, INDIRECT, or RLE.</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the type of index: NONE/FULL.</td>
</tr>
<tr>
<td>COLUMN_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the column.</td>
</tr>
<tr>
<td>PRELOAD</td>
<td>VARCHAR(5)</td>
<td>Displays if a column is preloaded: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_TEMPORARY_TABLES System View [page 2236]
- CREATE TABLE Statement (Data Definition) [page 825]
- TABLES System View [page 1669]
- TRUNCATE TABLE Statement (Data Manipulation) [page 1178]
- Handling Temporary Data
6.2.342 M_TEMPORARY_VIEWS System View

Displays temporary views.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view name.</td>
</tr>
<tr>
<td>VIEW_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the view.</td>
</tr>
<tr>
<td>IS_UNICODE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the query string contains Unicode: TRUE/FALSE.</td>
</tr>
<tr>
<td>IS_READ_ONLY</td>
<td>VARCHAR(5)</td>
<td>Displays whether this view is a read-only or an updatable view: TRUE/FALSE.</td>
</tr>
<tr>
<td>HAS_CHECK_OPTION</td>
<td>VARCHAR(5)</td>
<td>Displays whether this view has an updatable view condition: TRUE/FALSE.</td>
</tr>
<tr>
<td>HAS_COLUMN_ALIAS</td>
<td>VARCHAR(5)</td>
<td>Displays whether the view has a columns alias: TRUE/FALSE.</td>
</tr>
<tr>
<td>DEFINITION</td>
<td>NCLOB</td>
<td>Displays the definition of the view.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the description on the view.</td>
</tr>
<tr>
<td>IS_COLUMN_VIEW</td>
<td>VARCHAR(5)</td>
<td>Displays whether or not this view is a column view: TRUE/FALSE</td>
</tr>
<tr>
<td>VIEW_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of view: ROW, OLAP, JOIN, HIERARCHY, or CALC.</td>
</tr>
<tr>
<td>IS_TENANT_SHARED</td>
<td>VARCHAR(5)</td>
<td>Displays whether the view metadata can be shared across tenants: TRUE/FALSE</td>
</tr>
<tr>
<td>HAS_STRUCTURED_PRIVILEGE_CHECK</td>
<td>VARCHAR(5)</td>
<td>Displays whether the view is registered for a structured privilege check: TRUE/FALSE.</td>
</tr>
</tbody>
</table>
6.2.343 M_TEMPORARY_VIEW_COLUMNS System View

Provides information about temporary view columns.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the schema name.</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view name.</td>
</tr>
<tr>
<td>VIEW_OID</td>
<td>BIGINT</td>
<td>Displays the object ID of the view.</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the view column name.</td>
</tr>
<tr>
<td>POSITION</td>
<td>INTEGER</td>
<td>Displays the ordinal position of the view column.</td>
</tr>
<tr>
<td>DATA_TYPE_ID</td>
<td>SMALLINT</td>
<td>Displays the SQL data type ID of the column.</td>
</tr>
<tr>
<td>DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the SQL data type name of the column.</td>
</tr>
<tr>
<td>OFFSET</td>
<td>SMALLINT</td>
<td>Displays the offset of the column in a record.</td>
</tr>
<tr>
<td>LENGTH</td>
<td>INTEGER</td>
<td>Displays the number of:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Characters for char types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Max digits for numeric types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Characters for datetime types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Bytes for LOB types.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SCALE</td>
<td>INTEGER</td>
<td>Displays the maximum number of digits to the right of the decimal point: TIME or TIMESTAMP. The decimal digits are defined as the number of digits to the right of the decimal point in the second's component of the data.</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is allowed to accept null value: TRUE/FALSE.</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the default value in bytes.</td>
</tr>
<tr>
<td>COLLATION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the collation.</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>NVARCHAR(5000)</td>
<td>Displays the description for this column.</td>
</tr>
<tr>
<td>MAX_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>MIN_VALUE</td>
<td>NVARCHAR(5000)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>CS_DATA_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the column store data type ID.</td>
</tr>
<tr>
<td>CS_DATA_TYPE_NAME</td>
<td>VARCHAR(16)</td>
<td>Displays the column store data type name.</td>
</tr>
<tr>
<td>DDIC_DATA_TYPE_ID</td>
<td>INTEGER</td>
<td>Displays the DDIC data type ID.</td>
</tr>
<tr>
<td>DDIC_DATA_TYPE_NAME</td>
<td>VARCHAR(7)</td>
<td>Displays the DDIC data type name.</td>
</tr>
<tr>
<td>COMPRESSION_TYPE</td>
<td>VARCHAR(9)</td>
<td>Displays the type of compression: DEFAULT, PREFIXED, SPARSE, CLUSTERED, INDIRECT, or RLE.</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>VARCHAR(4)</td>
<td>Displays the type of index: NONE/FULL.</td>
</tr>
<tr>
<td>COLUMN_ID</td>
<td>BIGINT</td>
<td>Displays the ID of the column.</td>
</tr>
<tr>
<td>PRELOAD</td>
<td>VARCHAR(5)</td>
<td>Displays whether the column is pre-loaded: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

**Related Information**

M_TEMPORARY_VIEWS System View [page 2240]
M_TEMPORARY_TABLES System View [page 2236]
M_TEMPORARY_OBJECT_DEPENDENCIES System View [page 2234]
6.2.344  M_TEXT_ANALYSIS_LANGUAGES System View

Provides a list of supported languages.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANGUAGE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the language name.</td>
</tr>
<tr>
<td>LANGUAGE_CODE</td>
<td>VARCHAR(2)</td>
<td>Displays the language ISO 639-1 code.</td>
</tr>
</tbody>
</table>

**Related Information**

M_TEXT_ANALYSIS_MIME_TYPES System View [page 2243]
Introduction to SQL [page 27]
TEXT_CONFIGURATIONS System View [page 1691]

6.2.345  M_TEXT_ANALYSIS_MIME_TYPES System View

Provides a list of supported mime types.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIME_TYPE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the name of the mime type.</td>
</tr>
<tr>
<td>MIME_TYPE_DESCRIPTION</td>
<td>NVARCHAR(256)</td>
<td>Displays the description of the mime type.</td>
</tr>
</tbody>
</table>
6.2.346  **M_TIMEZONE_ALERTS System View**

Provides information about alerts relating to the status of SAP HANA internal timezone conversion.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(32)</td>
<td>Displays the status to report: TABLES NOT FOUND, TABLES EMPTY, NO SYSTEM TIMEZONE, SYSTEM TIMEZONE DISABLED, or CONVERSION MISMATCH.</td>
</tr>
<tr>
<td>TIMEZONE_NAME</td>
<td>NVARCHAR(64)</td>
<td>Displays the name of the timezone to which status message applies or NULL if the status does not apply to a specific timezone.</td>
</tr>
<tr>
<td>MISMATCH_BEGIN</td>
<td>TIMESTAMP</td>
<td>Displays the UTC time at which the first mismatch of the mismatch period occurred or NULL if no mismatch applies.</td>
</tr>
<tr>
<td>MISMATCH_END</td>
<td>TIMESTAMP</td>
<td>Displays the UTC time at which the last mismatch of the mismatch period occurred or NULL if no mismatch applies.</td>
</tr>
<tr>
<td>DETAILS</td>
<td>NVARCHAR(256)</td>
<td>Displays the detailed information.</td>
</tr>
</tbody>
</table>
6.2.347  M_TOPOLOGY_TREE System View

Provides information about SAP HANA nameserver topology content.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATH</td>
<td>NVARCHAR(1024)</td>
<td>Displays the path to the key.</td>
</tr>
<tr>
<td>NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the key name.</td>
</tr>
<tr>
<td>VALUE</td>
<td>NVARCHAR(256)</td>
<td>Displays the key value.</td>
</tr>
<tr>
<td>LEAF</td>
<td>NVARCHAR(5)</td>
<td>Displays the leaf flag: TRUE/FALSE.</td>
</tr>
</tbody>
</table>

Additional Information

Use '/' as the root path. For deeper paths, use PATH+'/'+NAME. If NAME contains '/', then you must enclose it in CHR(1). For example:

```
SELECT * FROM M_TOPOLOGY_TREE WHERE PATH='/index/'||CHR(1)||'SYSTEM:A/B'||CHR(1)||'15046'
```

To change topology tree values, use ALTER SYSTEM ALTER CONFIGURATION ('topology.ini','system') SET (PATH,NAME)=VALUE.

To delete values use ALTER SYSTEM ALTER CONFIGURATION ('topology.ini','system') UNSET (PATH,NAME).

Related Information

Partitioning Data Volumes
6.2.348 M_TRACEFILES System View

Provides information about all trace files.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the file name.</td>
</tr>
<tr>
<td>FILE_SIZE</td>
<td>BIGINT</td>
<td>Displays the file size in bytes.</td>
</tr>
<tr>
<td>FILE_MTIME</td>
<td>TIMESTAMP</td>
<td>Displays the file date.</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Shows the used disk space. This can help to identify issues caused by a full disk.</td>
</tr>
</tbody>
</table>

Additional Information

With the CLEAR command, all files that were opened by a service are removed or reset to size 0. On a distributed system, this command clears all traces on all hosts.

```sql
ALTER SYSTEM CLEAR TRACES
  ('ALERT', 'CLIENT', 'CRASHDUMP', '*', 'INDEXSERVER',..., 'DAEMON');
```

It can clear different types of files:

<table>
<thead>
<tr>
<th>Name.</th>
<th>Files.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALERT.</td>
<td>&lt;service&gt;alert....trc.</td>
</tr>
<tr>
<td>CLIENT.</td>
<td>localclient_....trc.</td>
</tr>
<tr>
<td>CRASHDUMP.</td>
<td>*.crashdump....trc.</td>
</tr>
<tr>
<td>*</td>
<td>open * .trc files of all active services.</td>
</tr>
</tbody>
</table>
**Related Information**

- **M_TRACEFILE_CONTENTS System View** [page 2247]
- **ALTER SYSTEM REMOVE TRACES Statement (System Management)** [page 564]
- **ALTER SYSTEM CLEAR TRACES Statement (System Management)** [page 525]
- **M_SERVICE_TRACES System View** [page 2132]

### 6.2.349 M_TRACEFILE_CONTENTS System View

Provides SAP HANA information from trace files.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the file name.</td>
</tr>
<tr>
<td>OFFSET</td>
<td>BIGINT</td>
<td>Displays the file offset in bytes.</td>
</tr>
<tr>
<td>CONTENT</td>
<td>NVARCHAR(1000)</td>
<td>Displays the file content at the offset.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view requires an equal predicate on HOST and FILE_NAME as part of the WHERE clause. Use HOST, FILE_NAME as returned by the view M_TRACEFILES to prevent unintentional materialization of all trace files.

The optional WHERE clause `OFFSET<,<=,>,>= value` is efficiently handled by this view. With `OFFSET > -value`, you can read from the end of the file without having to know the file size in advance. With `OFFSET <= -value`, you can read from the start and end of a file. Do not use the equivalent `OFFSET < value OR OFFSET > -value` because this is very inefficient for large files and returns duplicates for small files where the file size is < 2*value.
Trace files typically contain ASCII or CESU-8, but they can also contain binary data. To support all types of data, each byte from the trace file is encoded as one NVARCHAR with values in the range 0 to 255. To recode as CESU-8 you have to use code like in this Python example:

```python
cursor.execute("select CONTENT from M_TRACEFILE_CONTENTS where HOST='...' and FILE_NAME='...' ")
filedata=''
for row in cursor.fetchall():
    filedata += row[1].encode('latin-1') # reinterpret as bytearray
displaydata = filedata.decode('utf-8','replace') # do not use 'strict' error handling
```

### Related Information

- **M_TRACEFILES System View** [page 2246]
- **ALTER SYSTEM CLEAR TRACES Statement (System Management)** [page 525]
- **ALTER SYSTEM REMOVE TRACES Statement (System Management)** [page 564]
- **M_SERVICE_TRACES System View** [page 2132]

### 6.2.350 M_TRACE_CONFIGURATION System View

Provides information about trace configuration statistics.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the Persistence Volume ID.</td>
</tr>
<tr>
<td>STATISTICS_NAME</td>
<td>VARCHAR(128)</td>
<td>Displays the statistics object name.</td>
</tr>
<tr>
<td>TRACE_DIRECTORY</td>
<td>VARCHAR(512)</td>
<td>Displays the default trace level.</td>
</tr>
<tr>
<td>FORMATTER</td>
<td>VARCHAR(16)</td>
<td>Displays the trace formatter name.</td>
</tr>
<tr>
<td>FILE_NAME_PREFIX</td>
<td>VARCHAR(512)</td>
<td>Displays the trace file name prefix.</td>
</tr>
<tr>
<td>ALERT_FORMATTER</td>
<td>VARCHAR(16)</td>
<td>Displays the alert trace formatter name.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ALERT_FILE_NAME_PREFIX</td>
<td>VARCHAR(512)</td>
<td>Displays the alert trace file name prefix.</td>
</tr>
<tr>
<td>TRACE_FLUSH_INTERVAL</td>
<td>INTEGER</td>
<td>Displays the trace flush interval in seconds.</td>
</tr>
<tr>
<td>ALERT TRACELEVEL</td>
<td>VARCHAR(16)</td>
<td>The trace level for alert traces.</td>
</tr>
<tr>
<td>DEFAULT TRACELEVEL</td>
<td>VARCHAR(16)</td>
<td>Displays the default trace level.</td>
</tr>
<tr>
<td>TRACE BUFFER SIZE</td>
<td>INTEGER</td>
<td>Displays the trace buffer size in bytes.</td>
</tr>
<tr>
<td>MAX TRACE FILE SIZE</td>
<td>BIGINT</td>
<td>Displays the max trace file size in bytes.</td>
</tr>
<tr>
<td>MAX TRACE FILE ID</td>
<td>INTEGER</td>
<td>Displays the max trace file ID.</td>
</tr>
<tr>
<td>MAX TRACE FILES</td>
<td>INTEGER</td>
<td>Displays the max trace files to keep.</td>
</tr>
<tr>
<td>ALERT TRACE BUFFER SIZE</td>
<td>INTEGER</td>
<td>Displays the alert trace buffer size in bytes.</td>
</tr>
<tr>
<td>MAX ALERT TRACE FILE SIZE</td>
<td>BIGINT</td>
<td>Displays the max alert trace file size in bytes.</td>
</tr>
<tr>
<td>TRACE_COMPRESSION_INTERVAL</td>
<td>INTEGER</td>
<td>Displays the interval, in seconds, to check for trace files that need to be compressed.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view has a resettable counterpart; you can see the values since the last reset in the M_TRACE_CONFIGURATION_RESET system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_TRACE_CONFIGURATION_RESET;`.

**Related Information**

- M_TRACE_CONFIGURATION_RESET System View [page 2250]
- ALTER SYSTEM CLEAR TRACES Statement (System Management) [page 525]
- ALTER SYSTEM REMOVE TRACES Statement (System Management) [page 564]
- M_SERVICE_TRACES System View [page 2132]
6.2.351 M_TRACE_CONFIGURATION_RESET System View

Provides information about trace configuration statistics since the last reset.

This view contains values accumulated since the last reset of the main view M_TRACE_CONFIGURATION. Refer to M_TRACE_CONFIGURATION for information about the structure and use of this view.

In addition to the members mentioned in M_TRACE_CONFIGURATION, this view also contains a timestamp field, RESET_TIME, which indicates the last time the data was reset.

Related Information

M_TRACE_CONFIGURATION System View [page 2248]

6.2.352 M_TRANSACTIONS System View

Provides information about all transactions created by users or the database.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the connection ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the transaction object ID (this number is reused after a transaction is closed).</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID where the transaction has started.</td>
</tr>
<tr>
<td>PRIMARY_TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the primary transaction object ID for the external transaction.</td>
</tr>
<tr>
<td>TRANSACTION SEQUENCE</td>
<td>INTEGER</td>
<td>Displays the transaction sequence number in a transaction.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_ID</td>
<td>BIGINT</td>
<td>Displays the write transaction ID (this number is ever increasing).</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TRANSACTION_STATUS</td>
<td>VARCHAR(128)</td>
<td>Displays the transaction status; INACTIVE, ACTIVE, PRECOMMITTED, ABORTING, PARTIAL_ABORTING, or ACTIVE_PREPARE_COMMIT.</td>
</tr>
<tr>
<td>TRANSACTION_TYPE</td>
<td>VARCHAR(128)</td>
<td>Displays the transaction type; USER, VERSION GARBAGE COLLECTION, DDL VERSION GARBAGE COLLECTION, INTERNAL, or EXTERNAL.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the transaction start time.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the transaction end time.</td>
</tr>
<tr>
<td>START_MVCC_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the internal MVCC timestamp of the transaction start time.</td>
</tr>
<tr>
<td>END_MVCC_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the internal MVCC timestamp of the transaction end time.</td>
</tr>
<tr>
<td>EXECUTED_STATEMENT_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of executed statements in the transaction.</td>
</tr>
<tr>
<td>CREATED_VERSION_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of created versions in the transaction.</td>
</tr>
<tr>
<td>ALLOCATED_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the allocated version size in the transaction in bytes.</td>
</tr>
<tr>
<td>ACQUIRED_LOCK_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of acquired locks in the transaction.</td>
</tr>
<tr>
<td>LOCK_WAIT_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of lock waits in the transaction.</td>
</tr>
<tr>
<td>LOCK_WAIT_TIME</td>
<td>DOUBLE</td>
<td>Displays the accumulated lock wait time in the transaction in seconds.</td>
</tr>
<tr>
<td>LOCK_WAIT_TIMEOUT</td>
<td>INTEGER</td>
<td>Displays the number of milliseconds a statement waits for a lock to be released.</td>
</tr>
<tr>
<td>LOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the generated log amount in the transaction in bytes.</td>
</tr>
<tr>
<td>CURRENT_STATEMENT_ID</td>
<td>VARCHAR(256)</td>
<td>Displays the current statement ID.</td>
</tr>
<tr>
<td>CURRENT_STATEMENT_SEQUENCE</td>
<td>INTEGER</td>
<td>Displays the sequence number of the currently executed statement.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ALLOCATED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the memory size occupied by the transaction in bytes.</td>
</tr>
<tr>
<td>ACQUIRED_METALOCK_INDEX</td>
<td>INTEGER</td>
<td>Displays the index of the acquired metalock.</td>
</tr>
<tr>
<td>LOG_PARTITION_ID</td>
<td>SMALLINT</td>
<td>Displays the log partition ID of the transaction.</td>
</tr>
<tr>
<td>REDO_LOG_AMOUNT</td>
<td>BIGINT</td>
<td>Displays the size of the redo log amount generated by the transaction in bytes.</td>
</tr>
<tr>
<td>UNDO_LOG_AMOUNT</td>
<td>BIGINT</td>
<td>Displays the size of the undo log amount generated by the transaction in bytes.</td>
</tr>
<tr>
<td>MIN_MVCC_SNAPSHOT_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the logical timestamp of the first executed statement in the transaction.</td>
</tr>
<tr>
<td>LAST_COMMIT_ID</td>
<td>BIGINT</td>
<td>Displays the last commit ID of the transaction.</td>
</tr>
<tr>
<td>ACTIVE_STATEMENT_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of opened cursors in the transaction.</td>
</tr>
<tr>
<td>ISOLATION_LEVEL</td>
<td>VARCHAR(128)</td>
<td>Displays the isolation level of each transaction: READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, or SERIALIZABLE.</td>
</tr>
<tr>
<td>LOG_FLUSH_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether log flushing is enabled for the transaction (TRUE/FALSE).</td>
</tr>
<tr>
<td>LOGGING_ENABLED</td>
<td>VARCHAR(5)</td>
<td>Displays whether logging is enabled for the transaction (TRUE/FALSE).</td>
</tr>
</tbody>
</table>

**Related Information**

- Transaction Management Statements [page 1208]
- M_GARBAGE_COLLECTION_STATISTICS System View [page 1920]
- M_BLOCKED_TRANSACTIONS System View [page 1761]
- M_TRANS_TOKENS System View [page 2253]
- Workload in the Context of SAP HANA Autonomous Transaction
- COMMIT and ROLLBACK
6.2.353 M_TRANS_TOKENS System View

Provides information about all active transaction tokens.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>TRANS_TOKEN_ID</td>
<td>BIGINT</td>
<td>Displays the transaction token ID.</td>
</tr>
<tr>
<td>PRIMARY_TRANS_TOKEN_ID</td>
<td>BIGINT</td>
<td>Displays the primary transaction token ID for the shipped transaction token. If the transaction token is local-only, then this has the same value as TRANS_TOKEN_ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Displays the associated transaction object ID.</td>
</tr>
<tr>
<td>LOGICAL_CONNECTION_ID</td>
<td>INTEGER</td>
<td>Displays the associated logical connection ID.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>BIGINT</td>
<td>Displays the associated logical statement ID.</td>
</tr>
<tr>
<td>MVCC_SNAPSHOT_TIMESTAMP</td>
<td>BIGINT</td>
<td>Displays the MVCC timestamp that is held by the transaction token.</td>
</tr>
<tr>
<td>MVCC_SNAPSHOT_SCOPE</td>
<td>VARCHAR(8)</td>
<td>Displays the MVCC snapshot scope: TABLE/GLOBAL.</td>
</tr>
<tr>
<td>ES_SNAPSHOT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of extended storage snapshots.</td>
</tr>
<tr>
<td>ES_SNAPSHOTS</td>
<td>VARCHAR(1024)</td>
<td>Displays a list of volume and snapshot IDs for all extended storage snapshots.</td>
</tr>
</tbody>
</table>
6.2.354 M_UNDO_CLEANUP_FILES System View

Provides information about undo files and cleanup files.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the file type: UNDO, CLEANUP, or FREE.</td>
</tr>
<tr>
<td>TID</td>
<td>BIGINT</td>
<td>Displays the transaction ID.</td>
</tr>
<tr>
<td>PAGE_COUNT</td>
<td>BIGINT</td>
<td>Displays the page count.</td>
</tr>
<tr>
<td>RAW_SIZE</td>
<td>BIGINT</td>
<td>Displays the raw size in bytes.</td>
</tr>
<tr>
<td>CLEANUP_MARK</td>
<td>BIGINT</td>
<td>Displays the cleanup position mark.</td>
</tr>
<tr>
<td>NESTED_SESSION_ID</td>
<td>INTEGER</td>
<td>Displays the nested session ID.</td>
</tr>
<tr>
<td>NESTED_SESSION_PARENT_ID</td>
<td>INTEGER</td>
<td>Displays the nested session parent ID.</td>
</tr>
<tr>
<td>DEPENDENT_INDEX</td>
<td>INTEGER</td>
<td>Displays the dependent index during redo.</td>
</tr>
<tr>
<td>INDOUBT_FLAG</td>
<td>VARCHAR(5)</td>
<td>Displays the indoubt flag for distributed transaction: TRUE/FALSE.</td>
</tr>
<tr>
<td>TENTATIVE_PRECOMMIT_POSITION</td>
<td>BIGINT</td>
<td>Displays the tentative precommit position.</td>
</tr>
</tbody>
</table>
### Additional Information

Each undo or cleanup file in the system is represented by one row in this view. Undo files contain information needed for transaction rollback. These files are removed on transaction end. If data is deleted but must still be accessible because of MVCC isolation, then the corresponding information is written to cleanup files.

At the end of the transaction, cleanup files are passed to history management. Garbage collection uses the cleanup files to finally remove data. Undo files and cleanup files may be cached and reused because of performance issues. Cached files are assigned the type FREE. Undo files for row store are assigned the type EXTERNALUNDO.

### Related Information

- **M_GARBAGE_COLLECTION_STATISTICS System View [page 1920]**

### 6.2.355 M_VERSION_MEMORY System View

Provides information about memory use of the row-store Multiversion Concurrency Control (MVCC) manager.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMIT_ID</td>
<td>BIGINT</td>
<td>Displays the commit ID. This value is -1 for non-committed transactions.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>ALLOCATED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of memory allocated for row-store version space in bytes.</td>
</tr>
<tr>
<td>USED_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of memory actually used by row-store versions in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RECLAIMED_VERSION_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of memory reclaimed by row-store version garbage collection in bytes.</td>
</tr>
<tr>
<td>FREE_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of memory freed and reusable in row-store version space in bytes.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_SYSTEM_REPLICATION_MVCC_HISTORY System View [page 2196]
- CURRENT_MVCC_SNAPSHOT_TIMESTAMP Function (Datetime) [page 168]
- M_CS_MVCC System View [page 1826]
- M_MVCC_SNAPSHOTS System View [page 2029]
- M_MVCC_OVERVIEW System View [page 2027]
- M_MVCC_TABLES System View [page 2030]

**6.2.356 M_VOLUMES System View**

Provides information about the volumes used by SAP HANA servers.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>SERVICE_NAME</td>
<td>VARCHAR(32)</td>
<td>Displays the service name.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID.</td>
</tr>
<tr>
<td>SUBPATH</td>
<td>VARCHAR(512)</td>
<td>Displays the subpath appended to M_DISKS_PATH.</td>
</tr>
<tr>
<td>LIVECACHE_STORE</td>
<td>VARCHAR(10)</td>
<td>Displays whether the livecache is enabled.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>REMOVE_STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the remove progress.</td>
</tr>
</tbody>
</table>

Related Information

- M_VOLUME_SIZES System View [page 2282]
- M_VOLUME_FILES System View [page 2257]
- M_DATA_VOLUME_STATISTICS System View [page 1861]
- M_PERSISTENT_MEMORY_VOLUME_STATISTICS System View [page 2049]
- M_PERSISTENT_MEMORY_VOLUMES System View [page 2047]
- Partitioning Data Volumes
- Data and Log Volumes

6.2.357 M_VOLUME_FILES System View

Provides information about volume files.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>FILE_TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of file: DATA, LOG, and so on.</td>
</tr>
<tr>
<td>FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the file name.</td>
</tr>
<tr>
<td>USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of used data within the file in bytes.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Displays the total file size in bytes.</td>
</tr>
</tbody>
</table>
Additional Information

Displays information about files in the volume directories. All files in these directories are shown, but only registered files (files currently used by the database) have a file type. TOTAL_SIZE is the size as reported by the file system.

The meaning of USED_SIZE depends on the file type:

- **DATA**: Displays the size of used and shadow pages in the data volume file.
- **LOG**: Displays that the used size equals the TOTAL_SIZE.

Related Information

M_DATA_VOLUME_PARTITION_STATISTICS System View [page 1858]
M_DATA_VOLUME_STATISTICS System View [page 1861]
Data and Log Volumes

6.2.358 M_VOLUME_IO_DETAILED_STATISTICS System View

Provides detailed statistics about file access.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the file system path.</td>
</tr>
<tr>
<td>FILESYSTEM_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the file system type.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of contained files.</td>
</tr>
<tr>
<td>CONFIGURATION</td>
<td>VARCHAR(128)</td>
<td>Displays the configuration parameters.</td>
</tr>
<tr>
<td>MAX_IO_BUFFER_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum I/O buffer size in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>APPEND_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of appends.</td>
</tr>
<tr>
<td>ACTIVE_APPEND_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active appends.</td>
</tr>
<tr>
<td>MIN_APPEND_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of the appended data in bytes.</td>
</tr>
<tr>
<td>AVG_APPEND_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of the appended data in bytes.</td>
</tr>
<tr>
<td>MAX_APPEND_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of the appended data in bytes.</td>
</tr>
<tr>
<td>SUM_APPEND_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of the appended data in bytes.</td>
</tr>
<tr>
<td>MIN_APPEND_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for appends.</td>
</tr>
<tr>
<td>AVG_APPEND_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for appends.</td>
</tr>
<tr>
<td>MAX_APPEND_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for appends.</td>
</tr>
<tr>
<td>SUM_APPEND_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for appends.</td>
</tr>
<tr>
<td>WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of writes.</td>
</tr>
<tr>
<td>ACTIVE_WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active writes.</td>
</tr>
<tr>
<td>MIN_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of the written data in bytes.</td>
</tr>
<tr>
<td>AVG_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of the written data in bytes.</td>
</tr>
<tr>
<td>MAX_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of the written data in bytes.</td>
</tr>
<tr>
<td>SUM_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of written data.</td>
</tr>
<tr>
<td>MIN_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for writes.</td>
</tr>
<tr>
<td>AVG_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for writes.</td>
</tr>
<tr>
<td>MAX_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for writes.</td>
</tr>
<tr>
<td>SUM_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for writes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TRIGGER_ASYNC_WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of trigger asynchronous writes.</td>
</tr>
<tr>
<td>ACTIVE_TRIGGER_ASYNC_WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active trigger asynchronous writes.</td>
</tr>
<tr>
<td>MIN_TRIGGER_ASYNC_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of the trigger asynchronous write data in bytes.</td>
</tr>
<tr>
<td>AVG_TRIGGER_ASYNC_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of the trigger asynchronous write data in bytes.</td>
</tr>
<tr>
<td>MAX_TRIGGER_ASYNC_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of the trigger asynchronous write data in bytes.</td>
</tr>
<tr>
<td>SUM_TRIGGER_ASYNC_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of the trigger asynchronous write data in bytes.</td>
</tr>
<tr>
<td>MIN_TRIGGER_ASYNC_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for trigger asynchronous writes.</td>
</tr>
<tr>
<td>AVG_TRIGGER_ASYNC_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for trigger asynchronous writes.</td>
</tr>
<tr>
<td>MAX_TRIGGER_ASYNC_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for trigger asynchronous writes.</td>
</tr>
<tr>
<td>SUM_TRIGGER_ASYNC_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for trigger asynchronous writes.</td>
</tr>
<tr>
<td>READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of reads.</td>
</tr>
<tr>
<td>ACTIVE_READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active reads.</td>
</tr>
<tr>
<td>MIN_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of the read data in bytes.</td>
</tr>
<tr>
<td>AVG_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of the read data in bytes.</td>
</tr>
<tr>
<td>MAX_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of the read data in bytes.</td>
</tr>
<tr>
<td>SUM_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of the read data in bytes.</td>
</tr>
<tr>
<td>MIN_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for reads.</td>
</tr>
<tr>
<td>AVG_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for reads.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for reads.</td>
</tr>
<tr>
<td>SUM_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for reads.</td>
</tr>
<tr>
<td>TRIGGER_ASYNC_READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of trigger asynchronous reads.</td>
</tr>
<tr>
<td>ACTIVE_TRIGGER_ASYNC_READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active trigger asynchronous reads.</td>
</tr>
<tr>
<td>MIN_TRIGGER_ASYNC_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of the trigger asynchronous read data.</td>
</tr>
<tr>
<td>AVG_TRIGGER_ASYNC_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of the trigger asynchronous read data.</td>
</tr>
<tr>
<td>MAX_TRIGGER_ASYNC_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of the trigger asynchronous read data.</td>
</tr>
<tr>
<td>SUM_TRIGGER_ASYNC_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of the trigger asynchronous read data.</td>
</tr>
<tr>
<td>MIN_TRIGGER_ASYNC_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for trigger asynchronous reads.</td>
</tr>
<tr>
<td>AVG_TRIGGER_ASYNC_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for trigger asynchronous reads.</td>
</tr>
<tr>
<td>MAX_TRIGGER_ASYNC_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for trigger asynchronous reads.</td>
</tr>
<tr>
<td>SUM_TRIGGER_ASYNC_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for trigger asynchronous reads.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view shows detailed I/O statistics for various buffer sizes. Each buffer size is a maximum value. That is, a buffer of the size 4k actually means <= 4k and a buffer size of 16k means 4k < buffer size <= 16k. The I/O time is the total time from starting a request until it finishes, including enqueue time. All measured times are periods of time between enqueueing and finishing a request. The reported times contain wait times if the request cannot be started immediately. Enqueue time is measured separately to see how many requests are executed asynchronously and how many are synchronous or blocking. For overall I/O performance, see TOTAL_READ_WRITE_SIZE and TOTAL_IO_TIME in the M_VOLUME_IO_TOTAL_STATISTICS system view.

This view has a resettable counterpart; you can see the values since the last reset in the M_VOLUME_IO_DETAILED_STATISTICS_RESET system view. To reset the view, execute the following statement: ALTER SYSTEM RESET MONITORING VIEW SYS.M_VOLUME_IO_DETAILED_STATISTICS_RESET;
6.2.359 M_VOLUME_IO_DETAILED_STATISTICS_RESET System View

Provides detailed statistics about file access since the last reset.

This view contains the values that have accumulated since the last reset of the main view M_VOLUME_IO_DETAILED_STATISTICS. Refer to M_VOLUME_IO_DETAILED_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_VOLUME_IO_DETAILED_STATISTICS, this view also contains a timestamp field, RESET_TIME, that indicates the last time the data was reset.

Related Information

M_VOLUME_IO_DETAILED_STATISTICS System View [page 2258]

6.2.360 M_VOLUME_IO_PERFORMANCE_STATISTICS System View - Deprecated

Deprecated. Use M_VOLUME_IO_TOTAL_STATISTICS instead.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the filesystem path.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>FILESYSTEM_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the filesystem type.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of contained files.</td>
</tr>
<tr>
<td>MAX_IO_BUFFER</td>
<td>BIGINT</td>
<td>Displays the maximum I/O buffer size in bytes.</td>
</tr>
<tr>
<td>READ_SYNC</td>
<td>BIGINT</td>
<td>Displays the number of synchronous reads.</td>
</tr>
<tr>
<td>WRITE_SYNC</td>
<td>BIGINT</td>
<td>Displays the number of synchronous writes.</td>
</tr>
<tr>
<td>READ_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the number of read requests.</td>
</tr>
<tr>
<td>WRITE_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the number of write requests.</td>
</tr>
<tr>
<td>READ_COMPLETIONS</td>
<td>BIGINT</td>
<td>Displays the number of read completions.</td>
</tr>
<tr>
<td>FAILED_READS</td>
<td>BIGINT</td>
<td>Displays the number of failed reads.</td>
</tr>
<tr>
<td>WRITE_COMPLETIONS</td>
<td>BIGINT</td>
<td>Displays the number of write completions.</td>
</tr>
<tr>
<td>FAILED_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of failed writes.</td>
</tr>
<tr>
<td>FULL_RETRY_READS</td>
<td>BIGINT</td>
<td>Displays the number of full retry reads.</td>
</tr>
<tr>
<td>FULL_RETRY_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of full retry writes.</td>
</tr>
<tr>
<td>SHORT_READS</td>
<td>BIGINT</td>
<td>Displays the number of short reads.</td>
</tr>
<tr>
<td>SHORT_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of short writes.</td>
</tr>
<tr>
<td>DELAYED_READ_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the number of delayed read requests.</td>
</tr>
<tr>
<td>DELAYED_WRITE_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the number of delayed write requests.</td>
</tr>
<tr>
<td>DEQUEUED_DELAYED_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the number of dequeued delayed requests.</td>
</tr>
<tr>
<td>RESUBMITTED_DELAYED_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the number of resubmitted delayed requests.</td>
</tr>
<tr>
<td>RESUBMITTED_DELAYED_REQUESTS_DELAYED_AGAIN</td>
<td>BIGINT</td>
<td>Displays the number of resubmitted delayed requests delayed.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ITEMS_IN_DELAY_QUEUE</td>
<td>BIGINT</td>
<td>Displays the number of items in the delay queue.</td>
</tr>
<tr>
<td>MAX_ITEMS_IN_DELAY_QUEUE</td>
<td>BIGINT</td>
<td>Displays the maximum size of the delay queue in bytes.</td>
</tr>
<tr>
<td>LAST_READ_SYNC_TIME</td>
<td>BIGINT</td>
<td>Displays the last time for synchronous reads.</td>
</tr>
<tr>
<td>MAX_READ_SYNC_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for synchronous reads.</td>
</tr>
<tr>
<td>MIN_READ_SYNC_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for synchronous reads.</td>
</tr>
<tr>
<td>SUM_READ_SYNC_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for synchronous readings.</td>
</tr>
<tr>
<td>AVG_READ_SYNC_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for synchronous reads.</td>
</tr>
<tr>
<td>LAST_WRITE_SYNC_TIME</td>
<td>BIGINT</td>
<td>Displays the last time for synchronous writes.</td>
</tr>
<tr>
<td>MAX_WRITE_SYNC_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for synchronous writes.</td>
</tr>
<tr>
<td>MIN_WRITE_SYNC_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for synchronous writes.</td>
</tr>
<tr>
<td>SUM_WRITE_SYNC_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for synchronous writes.</td>
</tr>
<tr>
<td>AVG_WRITE_SYNC_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for synchronous writes.</td>
</tr>
<tr>
<td>LAST_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the last time for read events.</td>
</tr>
<tr>
<td>MAX_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for read events.</td>
</tr>
<tr>
<td>MIN_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for read events.</td>
</tr>
<tr>
<td>SUM_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for read events.</td>
</tr>
<tr>
<td>AVG_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for read events.</td>
</tr>
<tr>
<td>LAST_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the last time for write events.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for write events.</td>
</tr>
<tr>
<td>MIN_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for write events.</td>
</tr>
<tr>
<td>SUM_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for write events.</td>
</tr>
<tr>
<td>AVG_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for write events.</td>
</tr>
<tr>
<td>LAST_READ_ENQUEUE_TIME</td>
<td>BIGINT</td>
<td>Displays the last time for enqueuing read I/O events.</td>
</tr>
<tr>
<td>MAX_READ_ENQUEUE_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for enqueuing read I/O events.</td>
</tr>
<tr>
<td>MIN_READ_ENQUEUE_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for enqueuing read I/O events.</td>
</tr>
<tr>
<td>SUM_READ_ENQUEUE_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for enqueuing read I/O events.</td>
</tr>
<tr>
<td>AVG_READ_ENQUEUE_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for enqueuing read I/O events.</td>
</tr>
<tr>
<td>LAST_WRITE_ENQUEUE_TIME</td>
<td>BIGINT</td>
<td>Displays the last time for enqueuing write I/O events.</td>
</tr>
<tr>
<td>MAX_WRITE_ENQUEUE_TIME</td>
<td>BIGINT</td>
<td>Displays the maximum time for enqueuing write I/O events.</td>
</tr>
<tr>
<td>MIN_WRITE_ENQUEUE_TIME</td>
<td>BIGINT</td>
<td>Displays the minimum time for enqueuing write I/O events.</td>
</tr>
<tr>
<td>SUM_WRITE_ENQUEUE_TIME</td>
<td>BIGINT</td>
<td>Displays the total time for enqueuing write I/O events.</td>
</tr>
<tr>
<td>AVG_WRITE_ENQUEUE_TIME</td>
<td>BIGINT</td>
<td>Displays the average time for enqueuing write I/O events.</td>
</tr>
<tr>
<td>LAST_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the last size of read data in bytes.</td>
</tr>
<tr>
<td>MAX_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of read data in bytes.</td>
</tr>
<tr>
<td>MIN_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of read data in bytes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SUM_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of read data in bytes.</td>
</tr>
<tr>
<td>AVG_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of read data in bytes.</td>
</tr>
<tr>
<td>LAST_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the last size of written data in bytes.</td>
</tr>
<tr>
<td>MAX_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of written data in bytes.</td>
</tr>
<tr>
<td>MIN_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of written data in bytes.</td>
</tr>
<tr>
<td>SUM_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of written data in bytes.</td>
</tr>
<tr>
<td>AVG_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of written data in bytes.</td>
</tr>
<tr>
<td>LAST_READ_SYNC_SIZE</td>
<td>BIGINT</td>
<td>Displays the last size of synchronously read data in bytes.</td>
</tr>
<tr>
<td>MAX_READ_SYNC_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of synchronously read data in bytes.</td>
</tr>
<tr>
<td>MIN_READ_SYNC_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of synchronously read data in bytes.</td>
</tr>
<tr>
<td>SUM_READ_SYNC_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of synchronously read data size.</td>
</tr>
<tr>
<td>AVG_READ_SYNC_SIZE</td>
<td>BIGINT</td>
<td>Displays the average size of synchronously read data in bytes.</td>
</tr>
<tr>
<td>LAST_WRITE_SYNC_SIZE</td>
<td>BIGINT</td>
<td>Displays the last size of synchronously written data in bytes.</td>
</tr>
<tr>
<td>MAX_WRITE_SYNC_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum size of synchronously written data in bytes.</td>
</tr>
<tr>
<td>MIN_WRITE_SYNC_SIZE</td>
<td>BIGINT</td>
<td>Displays the minimum size of synchronously written data in bytes.</td>
</tr>
<tr>
<td>SUM_WRITE_SYNC_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of synchronously written data in bytes.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
AVG_WRITE_SYNC_SIZE | BIGINT | Displays the average size of synchronously written data in bytes.

### Additional Information

This view has a resettable counterpart; you can see the values since the last reset in the M_VOLUME_IO_PERFORMANCE_STATISTICS_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_VOLUME_IO_PERFORMANCE_STATISTICS_RESET;
```

Resetting this view implicitly resets M_VOLUME_IO_DETAILED_STATISTICS and M_VOLUME_IO_DETAILED/RETRY_STATISTICS.

The following table shows the differences between M_VOLUME_IO_PERFORMANCE_STATISTICS and M_VOLUME_IO_DETAILED/RETRY_STATISTICS. Columns that are only valid in M_VOLUME_IO_DETAILED/RETRY_STATISTICS are not shown.

<table>
<thead>
<tr>
<th>M_VOLUME_IO_PERFORMANCE_STATISTICS</th>
<th>M_VOLUME_IO_DETAILED_STATISTICS</th>
<th>M_VOLUME_IO_RETRY_STATISTICS</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>HOST</td>
<td>HOST</td>
<td>none</td>
</tr>
<tr>
<td>PORT</td>
<td>PORT</td>
<td>PORT</td>
<td>none</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>VOLUME_ID</td>
<td>VOLUME_ID</td>
<td>none</td>
</tr>
<tr>
<td>PATH</td>
<td>PATH</td>
<td>PATH</td>
<td>none</td>
</tr>
<tr>
<td>FILESYSTEM_TYPE</td>
<td>FILESYSTEM_TYPE</td>
<td>FILESYSTEM_TYPE</td>
<td>none</td>
</tr>
<tr>
<td>TYPE</td>
<td>TYPE</td>
<td>TYPE</td>
<td>none</td>
</tr>
<tr>
<td>MAX_IO_BUFFER</td>
<td>MAX_IO_BUFFER</td>
<td>MAX_IO_BUFFER</td>
<td>none</td>
</tr>
<tr>
<td>READ_SYNC</td>
<td>READ_COUNT + ACTIVE_READ_COUNT</td>
<td>derived</td>
<td></td>
</tr>
<tr>
<td>WRITE_SYNC</td>
<td>WRITE_COUNT + ACTIVE_WRITE_COUNT</td>
<td>derived</td>
<td></td>
</tr>
<tr>
<td>READ_REQUESTS</td>
<td>TRIGGER_ASYNC_READ_COUNT + ACTIVE_TRIGGER_ASYNC_READ_COUNT</td>
<td>derived</td>
<td></td>
</tr>
<tr>
<td>M_VOLUME_IO_PERFORMANCE_STATISTICS</td>
<td>M_VOLUME_IO_DETAILED_STATISTICS</td>
<td>M_VOLUME_IO_RETRY_STATISTICS</td>
<td>Change</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------</td>
<td>-----------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>WRITE_REQUESTS</td>
<td>TRIGGER_ASYNC_WRITE_COUNT + ACTIVE_TRIGGER_ASYNC_WRITE_COUNT</td>
<td>TRIGGER_ASYNC_WRITE_COUNT</td>
<td>derived</td>
</tr>
<tr>
<td>READ_COMPLETIONS</td>
<td>TRIGGER_ASYNC_READ_COUNT</td>
<td></td>
<td>renamed</td>
</tr>
<tr>
<td>FAILED_READS</td>
<td>FAILED_READS</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>WRITE_COMPLETIONS</td>
<td>TRIGGER_ASYNC_WRITE_COUNT</td>
<td></td>
<td>renamed</td>
</tr>
<tr>
<td>FAILED_WRITES</td>
<td>FAILED_WRITES</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>FULL_RETRY_READS</td>
<td>FULL_RETRY_READS</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>FULL_RETRY_WRITES</td>
<td>FULL_RETRY_WRITES</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>SHORT_READS</td>
<td>SHORT_READS</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>SHORT_WRITES</td>
<td>SHORT_WRITES</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>DELAYED_READ_REQUESTS</td>
<td></td>
<td>removed</td>
<td></td>
</tr>
<tr>
<td>DELAYED_WRITE_REQUESTS</td>
<td></td>
<td>removed</td>
<td></td>
</tr>
<tr>
<td>DEQUEUED_DELAYED_REQUESTS</td>
<td></td>
<td>removed</td>
<td></td>
</tr>
<tr>
<td>RESUBMITTED_DELAYED_REQUESTS</td>
<td></td>
<td>removed</td>
<td></td>
</tr>
<tr>
<td>RESUBMITTED_DELAYED_REQUESTS_DELAYED_AGAIN</td>
<td></td>
<td>removed</td>
<td></td>
</tr>
<tr>
<td>ITEMS_IN_DELAY_QUEUE</td>
<td></td>
<td>removed</td>
<td></td>
</tr>
<tr>
<td>MAX_ITEMS_IN_DELAY_QUEUE</td>
<td></td>
<td>removed</td>
<td></td>
</tr>
<tr>
<td>LAST_READ_SYNC_TIME</td>
<td></td>
<td>removed</td>
<td></td>
</tr>
<tr>
<td>MAX_READ_SYNC_TIME</td>
<td>MAX_READ_TIME</td>
<td>renamed</td>
<td></td>
</tr>
<tr>
<td>MIN_READ_SYNC_TIME</td>
<td>MIN_READ_TIME</td>
<td>renamed</td>
<td></td>
</tr>
<tr>
<td>M_VOLUME_IO_PERFORMANCE_STATISTICS</td>
<td>M_VOLUME_IO_DETAILED_STATISTICS</td>
<td>M_VOLUME_IO_RETRY_STATISTICS</td>
<td>Change</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------</td>
<td>----------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>SUM_READ_SYNC_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>AVG_READ_SYNC_TIME</td>
<td>AVG_READ_TIME</td>
<td></td>
<td>renamed</td>
</tr>
<tr>
<td>LAST_WRITE_SYNC_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>MAX_WRITE_SYNC_TIME</td>
<td>MAX_WRITE_TIME</td>
<td></td>
<td>renamed</td>
</tr>
<tr>
<td>MIN_WRITE_SYNC_TIME</td>
<td>MIN_WRITE_TIME</td>
<td></td>
<td>renamed</td>
</tr>
<tr>
<td>SUM_WRITE_SYNC_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>AVG_WRITE_SYNC_TIME</td>
<td>AVG_WRITE_TIME</td>
<td></td>
<td>renamed</td>
</tr>
<tr>
<td>LAST_READ_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>MAX_READ_TIME</td>
<td>MAX_READ_TIME</td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>MIN_READ_TIME</td>
<td>MIN_READ_TIME</td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>SUM_READ_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>AVG_READ_TIME</td>
<td>AVG_READ_TIME</td>
<td></td>
<td>renamed</td>
</tr>
<tr>
<td>LAST_WRITE_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>MAX_WRITE_TIME</td>
<td>MAX_WRITE_TIME</td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>MIN_WRITE_TIME</td>
<td>MIN_WRITE_TIME</td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>SUM_WRITE_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>AVG_WRITE_TIME</td>
<td>AVG_WRITE_TIME</td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>LAST_READ_ENQUEUE_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>MAX_READ_ENQUEUE_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>MIN_READ_ENQUEUE_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>SUM_READ_ENQUEUE_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>AVG_READ_ENQUEUE_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>LAST_WRITE_ENQUEUE_TIME</td>
<td></td>
<td></td>
<td>removed</td>
</tr>
<tr>
<td>M_VOLUME_IO_PERFORMANCE_STATISTICS</td>
<td>M_VOLUME_IO_DETAILED_STATISTICS</td>
<td>M_VOLUME_IO_RETRY_STATISTICS</td>
<td>Change</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------</td>
<td>-----------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>MAX_WRITE.Enqueue_TIME</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIN_WRITE.Enqueue_TIME</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUM_WRITE.Enqueue_TIME</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVG_WRITE.Enqueue_TIME</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAST_READ_SIZE</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX_READ_SIZE</td>
<td>MAX_READ_SIZE</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>MIN_READ_SIZE</td>
<td>MIN_READ_SIZE</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>SUM_READ_SIZE</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVG_READ_SIZE</td>
<td>AVG_READ_SIZE</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>LAST_WRITE_SIZE</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX_WRITE_SIZE</td>
<td>MAX_WRITE_SIZE</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>MIN_WRITE_SIZE</td>
<td>MIN_WRITE_SIZE</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>SUM_WRITE_SIZE</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVG_WRITE_SIZE</td>
<td>AVG_WRITE_SIZE</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>LAST_READ_SYNC_SIZE</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX_READ_SYNC_SIZE</td>
<td>MAX_READ_SIZE</td>
<td>renamed</td>
<td></td>
</tr>
<tr>
<td>MIN_READ_SYNC_SIZE</td>
<td>MIN_READ_SIZE</td>
<td>renamed</td>
<td></td>
</tr>
<tr>
<td>SUM_READ_SYNC_SIZE</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVG_READ_SYNC_SIZE</td>
<td>AVG_READ_SIZE</td>
<td>renamed</td>
<td></td>
</tr>
<tr>
<td>LAST_WRITE_SYNC_SIZE</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX_WRITE_SYNC_SIZE</td>
<td>MAX_WRITE_SIZE</td>
<td>renamed</td>
<td></td>
</tr>
<tr>
<td>MIN_WRITE_SYNC_SIZE</td>
<td>MIN_WRITE_SIZE</td>
<td>renamed</td>
<td></td>
</tr>
<tr>
<td>SUM_WRITE_SYNC_SIZE</td>
<td>removed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_VOLUME_IO_PERFORMANCE_STATISTICS</td>
<td>M_VOLUME_IO_DETAILED_STATISTICS</td>
<td>M_VOLUME_IO_RETRY_STATISTICS</td>
<td>Change</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------</td>
<td>-----------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>AVG_WRITE_SYNC_SIZE</td>
<td>AVG_WRITE_SIZE</td>
<td></td>
<td>renamed</td>
</tr>
</tbody>
</table>

**Related Information**

ALTER SYSTEM RESET MONITORING VIEW Statement (System Management) [page 565]
M_VOLUME_IO_PERFORMANCE_STATISTICS_RESET System View - Deprecated [page 2271]
M_VOLUME_IO_DETAILED_STATISTICS System View [page 2258]
M_VOLUME_IO_TOTAL_STATISTICS System View [page 2279]

### 6.2.361 M_VOLUME_IO_PERFORMANCE_STATISTICS_RESET

System View - Deprecated

Deprecated. Use M_VOLUME_IO_TOTAL_STATISTICS instead.

This view contains the values that have accumulated since the last reset of the main view M_VOLUME_IO_PERFORMANCE_STATISTICS. Refer to M_VOLUME_IO_PERFORMANCE_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_VOLUME_IO_PERFORMANCE_STATISTICS, this view also contains a timestamp field, RESET_TIME, that indicates the last time the data was reset.

**Related Information**

M_VOLUME_IO_PERFORMANCE_STATISTICS System View - Deprecated [page 2262]
M_VOLUME_IO_STATISTICS_RESET System View - Deprecated [page 2279]
6.2.362 M_VOLUME_IO_RETRY_STATISTICS System View

Provides file access retry statistics.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the file system path.</td>
</tr>
<tr>
<td>FILESYSTEM_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the file system type.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of contained files.</td>
</tr>
<tr>
<td>CONFIGURATION</td>
<td>VARCHAR(128)</td>
<td>Displays the configuration parameters.</td>
</tr>
<tr>
<td>MAX_IO_BUFFER_SIZE</td>
<td>BIGINT</td>
<td>Displays the maximum I/O buffer size in bytes.</td>
</tr>
<tr>
<td>TOTAL_APPEND_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of all appends.</td>
</tr>
<tr>
<td>FAILED_APPEND_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of failed appends.</td>
</tr>
<tr>
<td>TOTAL_WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of all writes.</td>
</tr>
<tr>
<td>FAILED_WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of failed writes.</td>
</tr>
<tr>
<td>SHORT_WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of short writes.</td>
</tr>
<tr>
<td>REQUESTS_WITH_SHORT_WRITES_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of requests with short writes.</td>
</tr>
<tr>
<td>AVG_SHORT_WRITES_PER_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the average number of short writes per request with short writes.</td>
</tr>
<tr>
<td>MAX_SHORT_WRITES_PER_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum number of short writes per request.</td>
</tr>
<tr>
<td>FULL_RETRY_WRITE_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of full retry writes.</td>
</tr>
<tr>
<td>REQUESTS_WITH_FULL_RETRY_WRITES_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of requests with full retry writes.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AVG_FULL_RETRY_WRITES_PER_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the average number of full retry writes per request with full retry writes.</td>
</tr>
<tr>
<td>MAX_FULL_RETRY_WRITES_PER_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum number of full retry writes per request.</td>
</tr>
<tr>
<td>TOTAL_READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of all reads.</td>
</tr>
<tr>
<td>FAILED_READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of failed reads.</td>
</tr>
<tr>
<td>SHORT_READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of short reads.</td>
</tr>
<tr>
<td>REQUESTS_WITH_SHORT_READS_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of requests with short reads.</td>
</tr>
<tr>
<td>AVG_SHORT_READS_PER_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the average number of short reads per request with short reads.</td>
</tr>
<tr>
<td>MAX_SHORT_READS_PER_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum number of short reads per request.</td>
</tr>
<tr>
<td>FULL_RETRY_READ_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of full retry reads.</td>
</tr>
<tr>
<td>REQUESTS_WITH_FULL_RETRY_READS_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of requests with full retry reads.</td>
</tr>
<tr>
<td>AVG_FULL_RETRY_READS_PER_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the average number of full retry reads per request with full retry reads.</td>
</tr>
<tr>
<td>MAX_FULL_RETRY_READS_PER_REQUEST_COUNT</td>
<td>BIGINT</td>
<td>Displays the maximum number of full retry reads per request.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view shows retry and error I/O statistics for various buffer sizes. Each buffer size is a maximum value, so a buffer size of 4k actually means <= 4k and a buffer size of 16k means 4k < buffer size <= 16k. For overall I/O performance, see TOTAL_READ_WRITE_SIZE and TOTAL_IO_TIME in the M_VOLUME_IO_TOTAL_STATISTICS system view.

This view has a resettable counterpart; you can see the values since the last reset in the M_VOLUME_IO_RETRY_STATISTICS_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_VOLUME_IO_RETRY_STATISTICS_RESET;
```
6.2.363 M_VOLUME_IO_RETRY_STATISTICS_RESET System View

Provides file access retry statistics since the last reset.

This view contains the values that have accumulated since the last reset of the main view M_VOLUME_IO_RETRY_STATISTICS. Refer to M_VOLUME_IO_RETRY_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_VOLUME_IO_RETRY_STATISTICS, this view also contains a timestamp field, RESET_TIME, that indicates the last time the data was reset.

Related Information

M_VOLUME_IO_RETRY_STATISTICS System View [page 2272]

6.2.364 M_VOLUME_IO_STATISTICS System View - Deprecated

Deprecated. Use M_VOLUME_IO_TOTAL_STATISTICS instead.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the file system path.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FILESYSTEM_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the file system type.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of contained files.</td>
</tr>
<tr>
<td>CONFIGURATION</td>
<td>VARCHAR(128)</td>
<td>Displays the configuration parameters.</td>
</tr>
<tr>
<td>OPEN_CALL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of open calls.</td>
</tr>
<tr>
<td>CLOSE_CALL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of close calls.</td>
</tr>
<tr>
<td>EXISTS_CALL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of exists calls.</td>
</tr>
<tr>
<td>REMOVE_CALL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of remove calls.</td>
</tr>
<tr>
<td>RESIZE_CALL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of resize calls.</td>
</tr>
<tr>
<td>SYNC_CALL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of sync calls.</td>
</tr>
<tr>
<td>GETSIZE_CALL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of get size calls.</td>
</tr>
<tr>
<td>GETMODIFICATION-TIME_CALL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of get modification time calls.</td>
</tr>
<tr>
<td>GETDEVICEID_CALL_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of get DeviceID calls.</td>
</tr>
<tr>
<td>ENQUEUED_WRITE_REQUESTS</td>
<td>BIGINT</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>BLOCKED_WRITE_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the number of blocked write requests.</td>
</tr>
<tr>
<td>MAX_BLOCKED_WRITE_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the maximum number of blocked write requests.</td>
</tr>
<tr>
<td>EOF_READS</td>
<td>BIGINT</td>
<td>Displays the number of EOF reads.</td>
</tr>
<tr>
<td>TOTAL_SYNC_READS</td>
<td>BIGINT</td>
<td>Displays the number of synchronous reads.</td>
</tr>
<tr>
<td>TOTAL_ASYNC_READS</td>
<td>BIGINT</td>
<td>Displays the number of asynchronous reads with waiting.</td>
</tr>
<tr>
<td>TOTAL_TRIGGER_ASYNC_READS</td>
<td>BIGINT</td>
<td>Displays the number of triggered asynchronous reads.</td>
</tr>
<tr>
<td>TRIGGER_ASYNC_READ_RATIO</td>
<td>DOUBLE</td>
<td>Displays the trigger-ratio of asynchronous reads.</td>
</tr>
<tr>
<td>TOTAL_SHORT_READS</td>
<td>BIGINT</td>
<td>Displays the number of reads that read less bytes than requested.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TOTAL_FULL_RETRY_READS</td>
<td>BIGINT</td>
<td>Displays the number of full retry reads.</td>
</tr>
<tr>
<td>TOTAL_FAILED_READS</td>
<td>BIGINT</td>
<td>Displays the number of failed reads.</td>
</tr>
<tr>
<td>TOTAL_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of read data in bytes.</td>
</tr>
<tr>
<td>TOTAL_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the total read time.</td>
</tr>
<tr>
<td>TOTAL_APPENDS</td>
<td>BIGINT</td>
<td>Displays the number of appends.</td>
</tr>
<tr>
<td>TOTAL_SYNC_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of synchronous writes.</td>
</tr>
<tr>
<td>TOTAL_ASYNC_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of asynchronous writes with waiting.</td>
</tr>
<tr>
<td>TOTAL_TRIGGER_ASYNC_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of triggered asynchronous writes.</td>
</tr>
<tr>
<td>TRIGGER_ASYNC_WRITE_RATIO</td>
<td>DOUBLE</td>
<td>Displays the trigger-ratio of asynchronous writes.</td>
</tr>
<tr>
<td>TOTAL_SHORT_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of writes that wrote less bytes than requested.</td>
</tr>
<tr>
<td>TOTAL_FULL_RETRY_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of full retry writes.</td>
</tr>
<tr>
<td>TOTAL_FAILED_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of failed writes.</td>
</tr>
<tr>
<td>TOTAL_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the amount of written data in bytes.</td>
</tr>
<tr>
<td>TOTAL_WRITE_TIME</td>
<td>BIGINT</td>
<td>Displays the total write time.</td>
</tr>
<tr>
<td>TOTAL_IO_TIME</td>
<td>BIGINT</td>
<td>Displays the total I/O time.</td>
</tr>
</tbody>
</table>

**Additional Information**

This view shows information about basic I/O operations on I/O subsystems (paths). TOTAL_READ/WRITE_SIZE is aggregated from M_VOLUME_IO_PERFORMANCE_STATISTICS. The column TOTAL_IO_TIME is only an approximation of real I/O times and should not be used to compute exact throughput ratios.

Refer to the M_VOLUME_IO_PERFORMANCE_STATISTICS system view for detailed information about read/write performance on various buffer sizes.
The following table shows the differences between M_VOLUME_IO_STATISTICS and M_VOLUME_IO_TOTAL_STATISTICS. Columns that are only valid in M_VOLUME_IO_TOTAL_STATISTICS are not shown.

<table>
<thead>
<tr>
<th>M_VOLUME_IO_STATISTICS</th>
<th>M_VOLUME_IO_TOTAL_STATISTICS</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>HOST</td>
<td>None</td>
</tr>
<tr>
<td>PORT</td>
<td>PORT</td>
<td>None</td>
</tr>
<tr>
<td>PATH</td>
<td>PATH</td>
<td>None</td>
</tr>
<tr>
<td>FILESYSTEM_TYPE</td>
<td>FILESYSTEM_TYPE</td>
<td>None</td>
</tr>
<tr>
<td>TYPE</td>
<td>TYPE</td>
<td>None</td>
</tr>
<tr>
<td>CONFIGURATION</td>
<td>CONFIGURATION</td>
<td>None</td>
</tr>
<tr>
<td>OPEN_CALL_COUNT</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>CLOSE_CALL_COUNT</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>EXISTS_CALL_COUNT</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>REMOVE_CALL_COUNT</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>RESIZE_CALL_COUNT</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>SYNC_CALL_COUNT</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>GETSIZE_CALL_COUNT</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>GETMODIFICATION_TIME_CALL_COUNT</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>GETDEVICEID_CALL_COUNT</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>ENQUEUED_WRITE_REQUESTS</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>BLOCKED_WRITE_REQUESTS</td>
<td>BLOCKED_WRITE_REQUESTS</td>
<td>None</td>
</tr>
<tr>
<td>MAX_BLOCKED_WRITE_REQUESTS</td>
<td>MAX_BLOCKED_WRITE_REQUESTS</td>
<td>None</td>
</tr>
<tr>
<td>EOF_READS</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>TOTAL_SYNC_READS</td>
<td>TOTAL_READS</td>
<td>Renamed</td>
</tr>
<tr>
<td>TOTAL_ASYNC_READS</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>TOTAL_TRIGGER_ASYNC_READS</td>
<td>TOTAL_TRIGGER_ASYNC_READS</td>
<td>None</td>
</tr>
<tr>
<td>TRIGGER_ASYNC_READ_RATIO</td>
<td>TRIGGER_READ_RATIO</td>
<td>Renamed</td>
</tr>
<tr>
<td>M_VOLUME_IO_STATISTICS</td>
<td>M_VOLUME_IO_TOTAL_STATISTICS</td>
<td>Change</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>TOTAL_SHORT_READS</td>
<td>TOTAL_SHORT_READS</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_FULL_RETRY_READS</td>
<td>TOTAL_FULL_RETRY_READS</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_FAILED_READS</td>
<td>TOTAL_FAILED_READS</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_READ_SIZE</td>
<td>TOTAL_READ_SIZE</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_READ_TIME</td>
<td>TOTAL_READ_TIME</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_APPENDS</td>
<td>TOTAL_APPENDS</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_SYNC_WRITES</td>
<td>TOTAL_WRITES</td>
<td>Renamed</td>
</tr>
<tr>
<td>TOTAL_ASYNC_WRITES</td>
<td></td>
<td>Removed</td>
</tr>
<tr>
<td>TOTAL_TRIGGER_ASYNC_WRITES</td>
<td>TOTAL_TRIGGER_ASYNC_WRITES</td>
<td>None</td>
</tr>
<tr>
<td>TRIGGER_ASYNC_WRITE_RATIO</td>
<td>TRIGGER_WRITE_RATIO</td>
<td>Renamed</td>
</tr>
<tr>
<td>TOTAL_SHORT_WRITES</td>
<td>TOTAL_SHORT_WRITES</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_FULL_RETRY_WRITES</td>
<td>TOTAL_FULL_RETRY_WRITES</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_FAILED_WRITES</td>
<td>TOTAL_FAILED_WRITES</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_WRITE_SIZE</td>
<td>TOTAL_WRITE_SIZE</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_WRITE_TIME</td>
<td>TOTAL_WRITE_TIME</td>
<td>None</td>
</tr>
<tr>
<td>TOTAL_IO_TIME</td>
<td>TOTAL_IO_TIME</td>
<td>None</td>
</tr>
</tbody>
</table>

This view has a resettable counterpart; you can see the values since the last reset in the M_VOLUME_IO_STATISTICS_RESET system view. To reset the view, execute the following statement: `ALTER SYSTEM RESET MONITORING VIEW SYS.M_VOLUME_IO_STATISTICS_RESET;`.

Resetting this view implicitly resets the dependent child views M_VOLUME_IO_DETAILED_STATISTICS and M_VOLUME_IO_RETRY_STATISTICS.

**Related Information**

- ALTER SYSTEM RESET MONITORING VIEW Statement (System Management) [page 565]
- M_VOLUME_IO_STATISTICS_RESET System View - Deprecated [page 2279]
- M_VOLUME_IO_DETAILED_STATISTICS System View [page 2258]
- M_VOLUME_IO_TOTAL_STATISTICS System View [page 2279]
- M_VOLUME_IO_STATISTICS System View - Deprecated [page 2274]
6.2.365  M_VOLUME_IO_STATISTICS_RESET System View - Deprecated

Deprecated. Use M_VOLUME_IO_TOTAL_STATISTICS instead.

This view contains the values that have accumulated since the last reset of the main view M_VOLUME_IO_STATISTICS. Refer to M_VOLUME_IO_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_VOLUME_IO_STATISTICS, this view also contains a timestamp field, RESET_TIME, that indicates the last time the data was reset.

Related Information

M_VOLUME_IO_STATISTICS System View - Deprecated [page 2274]

6.2.366  M_VOLUME_IO_TOTAL_STATISTICS System View

Shows information about basic I/O operations on I/O subsystems (paths).

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port number.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the persistence volume ID.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Displays the file system path.</td>
</tr>
<tr>
<td>FILESYSTEM_TYPE</td>
<td>VARCHAR(32)</td>
<td>Displays the file system type.</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(16)</td>
<td>Displays the type of contained files.</td>
</tr>
<tr>
<td>CONFIGURATION</td>
<td>VARCHAR(256)</td>
<td>Displays the configuration parameters.</td>
</tr>
<tr>
<td>BLOCKED_WRITE_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the number of blocked write requests.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_BLOCKED_WRITE_REQUESTS</td>
<td>BIGINT</td>
<td>Displays the maximum number of blocked write requests.</td>
</tr>
<tr>
<td>TOTAL_READS</td>
<td>BIGINT</td>
<td>Displays the number of synchronous reads.</td>
</tr>
<tr>
<td>TOTAL_TRIGGER_ASYNC_READS</td>
<td>BIGINT</td>
<td>Displays the number of triggered asynchronous reads.</td>
</tr>
<tr>
<td>ACTIVE_ASYNC_READS_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active asynchronous reads.</td>
</tr>
<tr>
<td>TRIGGER_READ_RATIO</td>
<td>DOUBLE</td>
<td>Displays the trigger-ratio of asynchronous reads.</td>
</tr>
<tr>
<td>TOTAL_SHORT_READS</td>
<td>BIGINT</td>
<td>Displays the number of reads that read fewer bytes than requested.</td>
</tr>
<tr>
<td>TOTAL_FULL_RETRY_READS</td>
<td>BIGINT</td>
<td>Displays the number of full retry reads.</td>
</tr>
<tr>
<td>TOTAL_FAILED_READS</td>
<td>BIGINT</td>
<td>Displays the number of failed reads.</td>
</tr>
<tr>
<td>TOTAL_READ_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of read data in bytes.</td>
</tr>
<tr>
<td>TOTAL_READ_TIME</td>
<td>BIGINT</td>
<td>Displays the total read time in microseconds.</td>
</tr>
<tr>
<td>TOTAL_APPENDS</td>
<td>BIGINT</td>
<td>Displays the number of appends.</td>
</tr>
<tr>
<td>TOTAL_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of synchronous writes.</td>
</tr>
<tr>
<td>TOTAL_TRIGGER_ASYNC_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of triggered asynchronous writes.</td>
</tr>
<tr>
<td>ACTIVE_ASYNC_WRITES_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of active asynchronous writes.</td>
</tr>
<tr>
<td>TRIGGER_WRITE_RATIO</td>
<td>DOUBLE</td>
<td>Displays the trigger-ratio of asynchronous writes.</td>
</tr>
<tr>
<td>TOTAL_SHORT_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of writes that wrote less bytes than requested.</td>
</tr>
<tr>
<td>TOTAL_FULL_RETRY_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of full retry writes.</td>
</tr>
<tr>
<td>TOTAL_FAILED_WRITES</td>
<td>BIGINT</td>
<td>Displays the number of failed writes.</td>
</tr>
<tr>
<td>TOTAL_WRITE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of written data in bytes.</td>
</tr>
</tbody>
</table>
### Column Name | Data Type | Description
---|---|---
TOTAL_WRITE_TIME | BIGINT | Displays the total write time in microseconds.
TOTAL_IO_TIME | BIGINT | Displays the total IO time in microseconds.

### Additional Information

Throughput of file I/O can generally be calculated by \((\text{TOTAL_READ_SIZE} + \text{TOTAL_WRITE_SIZE}) / \text{TOTAL_IO_TIME}\).

For TOTAL_IO_TIME, note that reads and writes can happen in parallel. For example, assume there is a 20 MB read operation and, in parallel, a 20 MB write operation. The read operation starts at 0.0 sec, the write operation starts at 0.5 sec, and both operations last exactly 2 seconds. TOTAL_IO_TIME is calculated as:

| TOTAL_READ_SIZE | 20,971,520 Bytes |
| TOTAL_WRITE_SIZE | 20,971,520 Bytes |
| TOTAL_READ_TIME | 2,000,000 Microsecond (timeframe: 0.0 - 2.0 sec) |
| TOTAL_WRITE_TIME | 2,000,000 Microsecond (timeframe: 0.5 - 2.5 sec) |
| TOTAL_IO_TIME | 2,500,000 Microsecond (timeframe: 0.0 - 2.5 sec) |

Refer to the M_VOLUME_IO_DETAILED_STATISTICS and M_VOLUME_IO_RETRY_STATISTICS system views for detailed information about read/write performance on various buffer sizes.

This view has a resettatable counterpart; you can see the values since the last reset in the M_VOLUME_IO_TOTAL_STATISTICS_RESET system view. To reset the view, execute the following statement:

```
ALTER SYSTEM RESET MONITORING VIEW SYS.M_VOLUME_IO_TOTAL_STATISTICS_RESET;
```

Resetting this view implicitly resets the dependent child views M_VOLUME_IO_DETAILED_STATISTICS and M_VOLUME_IO_RETRY_STATISTICS.

### Related Information

- M_VOLUME_IO_DETAILED_STATISTICS System View [page 2258]
- M_VOLUME_IO_RETRY_STATISTICS System View [page 2272]
- M_VOLUME_IO_DETAILED_STATISTICS System View [page 2258]
6.2.367  M_VOLUME_IO_TOTAL_STATISTICS_RESET System View

Provides file access statistics since the last reset.

This view contains the values that have been accumulated since the last reset of the main view M_VOLUME_IO_TOTAL_STATISTICS. Refer to M_VOLUME_IO_TOTAL_STATISTICS for information about the structure and use of this view.

In addition to the members mentioned in M_VOLUME_IO_TOTAL_STATISTICS, this view also contains a timestamp field, RESET_TIME, that indicates the last time the data was reset.

Related Information

M_VOLUME_IO_TOTAL_STATISTICS System View [page 2279]

6.2.368  M_VOLUME_SIZES System View

Provides information about volume sizes used by SAP HANA servers.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Displays the volume ID. See M_VOLUMES.</td>
</tr>
<tr>
<td>DISK_ID</td>
<td>INTEGER</td>
<td>Displays the disk ID. See M_DISKS.</td>
</tr>
<tr>
<td>DATA_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of files in the data directory, in bytes. For non-aggregated size, see M_VOLUME_FILES.</td>
</tr>
<tr>
<td>LOG_SIZE</td>
<td>BIGINT</td>
<td>Displays the total size of files in the log directory, in bytes. For non-aggregated size, see M_VOLUME_FILES.</td>
</tr>
<tr>
<td>DATA_USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total used disk space in the data directory, in bytes.</td>
</tr>
<tr>
<td>LOG_USED_SIZE</td>
<td>BIGINT</td>
<td>Displays the total used disk space in the log directory, in bytes.</td>
</tr>
</tbody>
</table>
Related Information

M_DISKS System View [page 1871]
M_VOLUMES System View [page 2256]

6.2.369 M_WORKLOAD System View

Provides information about the database workload collected every minute.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>EXECUTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the total count of all executed statements for data manipulation, data definition, and system control.</td>
</tr>
<tr>
<td>COMPILATION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of statement preparation.</td>
</tr>
<tr>
<td>UPDATE_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of update transactions.</td>
</tr>
<tr>
<td>COMMIT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of transaction commits.</td>
</tr>
<tr>
<td>ROLLBACK_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of transaction rollbacks.</td>
</tr>
<tr>
<td>CURRENT_EXECUTION_RATE</td>
<td>DOUBLE</td>
<td>Displays the current statement execution count per minute.</td>
</tr>
<tr>
<td>PEAK_EXECUTION_RATE</td>
<td>DOUBLE</td>
<td>Displays the peak statement execution count per minute.</td>
</tr>
<tr>
<td>CURRENT_COMPILATION_RATE</td>
<td>DOUBLE</td>
<td>Displays the current statement preparation count per minute.</td>
</tr>
<tr>
<td>PEAK_COMPILATION_RATE</td>
<td>DOUBLE</td>
<td>Displays the peak statement preparation count per minute.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CURRENT_UPDATE_TRANSACTION_RATE</td>
<td>DOUBLE</td>
<td>Displays the current update transaction count per minute.</td>
</tr>
<tr>
<td>PEAK_UPDATE_TRANSACTION_RATE</td>
<td>DOUBLE</td>
<td>Displays the peak update transaction count per minute.</td>
</tr>
<tr>
<td>CURRENT_TRANSACTION_RATE</td>
<td>DOUBLE</td>
<td>Displays the current transaction count per minute.</td>
</tr>
<tr>
<td>PEAK_TRANSACTION_RATE</td>
<td>DOUBLE</td>
<td>Displays the peak transaction count per minute.</td>
</tr>
<tr>
<td>CURRENT_COMMIT_RATE</td>
<td>DOUBLE</td>
<td>Displays the total number of commits per minute.</td>
</tr>
<tr>
<td>PEAK_COMMIT_RATE</td>
<td>DOUBLE</td>
<td>Displays the peak commit counts per minute.</td>
</tr>
<tr>
<td>CURRENT_ROLLBACK_RATE</td>
<td>DOUBLE</td>
<td>Displays the total number of rollbacks per minute.</td>
</tr>
<tr>
<td>PEAK_ROLLBACK_RATE</td>
<td>DOUBLE</td>
<td>Displays the peak rollback count per minute.</td>
</tr>
<tr>
<td>CURRENT_MEMORY_USAGE_RATE</td>
<td>DOUBLE</td>
<td>Displays the total size of used memory per minute in bytes.</td>
</tr>
<tr>
<td>PEAK_MEMORY_USAGE_RATE</td>
<td>DOUBLE</td>
<td>Displays the peak size of used memory per minute in bytes.</td>
</tr>
</tbody>
</table>

**Related Information**

- M_WORKLOAD_CAPTURES System View [page 2285]
- M_WORKLOAD_REPLAYS System View [page 2288]
- M_WORKLOAD_REPLAY_PREPROCESSES System View [page 2290]
- CREATE WORKLOAD CLASS Statement (Workload Management) [page 929]
- ALTER WORKLOAD CLASS Statement (Workload Management) [page 678]
- ALTER WORKLOAD MAPPING Statement (Workload Management) [page 681]
- CREATE WORKLOAD MAPPING Statement (Workload Management) [page 934]
- DROP WORKLOAD CLASS Statement (Workload Management) [page 989]
- DROP WORKLOAD MAPPING Statement (Workload Management) [page 990]
- Workload Management
# 6.2.370 M_WORKLOAD_CAPTURES System View

Provides information about workload captures.

## Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPTURE_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the user-specified name of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_DESCRIPTION</td>
<td>NVARCHAR (5000)</td>
<td>Displays the user-specified description of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_VERSION</td>
<td>INTEGER</td>
<td>Displays the version number of the structure format for the captured workloads or capturing workload.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start timestamp of the capture.</td>
</tr>
<tr>
<td>UTC_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the UTC start timestamp of the capture.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the end timestamp of the captured workload.</td>
</tr>
<tr>
<td>UTC_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the UTC end timestamp of the captured workload.</td>
</tr>
<tr>
<td>IS_FILTER_APPLIED</td>
<td>VARCHAR (5)</td>
<td>Displays the flag that is set to FALSE if the user Displays any non-default value to any filter parameters.</td>
</tr>
<tr>
<td>SYSTEM_ID</td>
<td>VARCHAR (3)</td>
<td>Displays the system ID in which the capture has been done.</td>
</tr>
<tr>
<td>DATABASE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the database name in which the capture has been done.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR (16)</td>
<td>Displays the status of the workload captures.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IS_LOADED</td>
<td>VARCHAR(5)</td>
<td>Displays the flag that is set to TRUE if the captured data is loaded into the database.</td>
</tr>
<tr>
<td>PROGRESS</td>
<td>DOUBLE</td>
<td>Displays the current progress of the load.</td>
</tr>
<tr>
<td>LEVEL</td>
<td>VARCHAR(16)</td>
<td>Displays the level of workload captures.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>NCLOB</td>
<td>Displays the applied parameter’s key/value pairs.</td>
</tr>
<tr>
<td>STATEMENT_DURATION_THRESHOLD</td>
<td>BIGINT</td>
<td>Displays the threshold value for the workloads to be captured.</td>
</tr>
<tr>
<td>FLUSH_INTERVAL</td>
<td>INTEGER</td>
<td>Displays the interval of flushing out the file output stream.</td>
</tr>
<tr>
<td>CAPTURE_FILE_NAME</td>
<td>VARCHAR(256)</td>
<td>Displays the file name of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_SESSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of current or captured logical sessions.</td>
</tr>
<tr>
<td>CAPTURE_STATEMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of current or captured statements.</td>
</tr>
<tr>
<td>CAPTURE_FETCH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number current or captured fetch operations.</td>
</tr>
<tr>
<td>CAPTURE_COMMITTED_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of current or captured committed transactions.</td>
</tr>
<tr>
<td>CAPTURE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of the captured workload file.</td>
</tr>
<tr>
<td>CAPTURE_FAILED_STATEMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of failed current or capture statements.</td>
</tr>
<tr>
<td>CAPTURE_FAILED_FETCH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of failed current or capture fetch operations.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the error code generated by the system during capturing.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>NVARCHAR(5000)</td>
<td>Displays the error message generated by system during capturing.</td>
</tr>
</tbody>
</table>
6.2.371 M_WORKLOAD_CLASS_STATISTICS System View

Displays how many resources have been consumed by statements that belong to a specified workload class.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>NVARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>WORKLOAD_CLASS_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the workload class name.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>Displays the total peak memory size, in bytes, used by statements.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_CPU_TIME</td>
<td>BIGINT</td>
<td>Displays the total CPU time, in microseconds, used by statements.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_ADMIN_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of admitted statements.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT_REJECT_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of rejected statements.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT.Enqueue_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of enqueued statements.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT.Dequeue_COUNT</td>
<td>BIGINT</td>
<td>Displays the total number of dequeued statements for deferred execution.</td>
</tr>
<tr>
<td>TOTAL_STATEMENT.Timeout_Count</td>
<td>BIGINT</td>
<td>Displays the total number of dequeued statements for timed out.</td>
</tr>
</tbody>
</table>
Related Information

WORKLOAD_CLASSES System View [page 1721]
M_WORKLOAD System View [page 2283]
M_WORKLOAD_CAPTURES System View [page 2285]
M_WORKLOAD_REPLAYS System View [page 2288]
M_WORKLOAD_REPLAY_PREPROCESSES System View [page 2290]
WORKLOAD_MAPPINGS System View [page 1723]
Managing Workload with Workload Classes
Monitoring Views for Workload Classes
Workload Class Examples

6.2.372 M_WORKLOAD_REPLAYS System View

Provides information about workload replays.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPTURE_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_SYSTEM_ID</td>
<td>VARCHAR(3)</td>
<td>Displays the system ID in which the capture has been done.</td>
</tr>
<tr>
<td>CAPTURE_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the database in which the capture has been done.</td>
</tr>
<tr>
<td>REPLAY_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of the replayed workload.</td>
</tr>
<tr>
<td>REPLAY_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user-specified name of the replayed workload.</td>
</tr>
<tr>
<td>REPLAY_DESCRIPTION</td>
<td>NVARCHAR(5000)</td>
<td>Displays the user-specified description of the replayed workload.</td>
</tr>
<tr>
<td>REPLAY_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start timestamp of the replay.</td>
</tr>
<tr>
<td>REPLAY_UTC_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the UTC start timestamp of the replay.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REPLAY_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the end timestamp of the replayed workload.</td>
</tr>
<tr>
<td>REPLAY_UTC_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the UTC end timestamp of the replayed workload.</td>
</tr>
<tr>
<td>REPLAY_HOST</td>
<td>VARCHAR (64)</td>
<td>Displays the host name in which the replay has been done.</td>
</tr>
<tr>
<td>REPLAY_PORT</td>
<td>INTEGER</td>
<td>Displays the service port number in which the replay has been done.</td>
</tr>
<tr>
<td>REPLAY_SYSTEM_ID</td>
<td>VARCHAR(3)</td>
<td>Displays the system ID in which the replay has been done.</td>
</tr>
<tr>
<td>REPLAY_DATABASE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the database in which the replay has been done.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>Displays the status of workload replays.</td>
</tr>
<tr>
<td>PROGRESS</td>
<td>DOUBLE</td>
<td>Displays a measure of how much the replayer has made its progress.</td>
</tr>
<tr>
<td>REPLAY_SESSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of current or replayed logical sessions.</td>
</tr>
<tr>
<td>REPLAY_STATEMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of current or replayed statements.</td>
</tr>
<tr>
<td>REPLAY_FETCH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of current or replayed fetch operations.</td>
</tr>
<tr>
<td>REPLAY_COMMITTED_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of current or replayed committed transactions.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the error code generated by the system during capturing.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>NVARCHAR (5000)</td>
<td>Displays the error message generated by the system during capturing.</td>
</tr>
</tbody>
</table>

**Related Information**

CREATE WORKLOAD CLASS Statement (Workload Management) [page 929]
ALTER WORKLOAD CLASS Statement (Workload Management) [page 678]
ALTER WORKLOAD MAPPING Statement (Workload Management) [page 681]
CREATE WORKLOAD MAPPING Statement (Workload Management) [page 934]
6.2.373 M_WORKLOAD_REPLAY_PREPROCESSES System View

Provides information about preprocesses for captured workloads.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPTURE_ID</td>
<td>BIGINT</td>
<td>Displays the unique ID of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the user-specified name of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_DESCRIPTION</td>
<td>NVARCHAR (5000)</td>
<td>Displays the user-specified description of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start timestamp of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_UTC_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the UTC start timestamp of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the end timestamp of the captured workload.</td>
</tr>
<tr>
<td>CAPTURE_UTC_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the UTC end timestamp of the captured workload.</td>
</tr>
<tr>
<td>IS_FILTER_APPLIED</td>
<td>VARCHAR (5)</td>
<td>Displays the flag to be set to 'false' if user specified any non-default value to any filter parameters.</td>
</tr>
<tr>
<td>CAPTURE_SYSTEM_ID</td>
<td>VARCHAR (3)</td>
<td>Displays the system ID, in which the capture has been done.</td>
</tr>
<tr>
<td>CAPTURE_DATABASE_NAME</td>
<td>NVARCHAR (256)</td>
<td>Displays the database name, in which the capture has been done.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CAPTURE_VERSION</td>
<td>INTEGER</td>
<td>Displays the version number of the structure format of the captured workloads or capturing workload.</td>
</tr>
<tr>
<td>PREPROCESS_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the start timestamp of the preprocessing.</td>
</tr>
<tr>
<td>PREPROCESS.Utc_START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the UTC start timestamp of the preprocessing.</td>
</tr>
<tr>
<td>PREPROCESS_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the end timestamp of the preprocessing.</td>
</tr>
<tr>
<td>PREPROCESS.Utc_END_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the UTC end timestamp of the preprocessing.</td>
</tr>
<tr>
<td>PREPROCESS_VERSION</td>
<td>INTEGER</td>
<td>Displays the version number of the structure format of the preprocessed workloads or preprocessing workload.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR (16)</td>
<td>Displays the status of workload preprocessing.</td>
</tr>
<tr>
<td>PROGRESS</td>
<td>DOUBLE</td>
<td>Displays the current progress of preprocessing.</td>
</tr>
<tr>
<td>LEVEL</td>
<td>VARCHAR (16)</td>
<td>Displays the level of workload captures.</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>NCLOB</td>
<td>Displays the applied parameter’s key/value pairs.</td>
</tr>
<tr>
<td>STATEMENT_DURATION_THRESHOLD</td>
<td>BIGINT</td>
<td>Displays the threshold value for the elapsed time of executed statements to be captured.</td>
</tr>
<tr>
<td>FLUSH_INTERVAL</td>
<td>INTEGER</td>
<td>Displays the interval of flushing out the file output stream.</td>
</tr>
<tr>
<td>CAPTURE_SESSION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of current or captured logical sessions.</td>
</tr>
<tr>
<td>CAPTURE_STATEMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of current or captured statements.</td>
</tr>
<tr>
<td>CAPTURE_FETCH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number current or captured fetch operations.</td>
</tr>
<tr>
<td>CAPTURE_COMMITTED_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of current or captured committed transactions.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CAPTURE_FAILED_STATEMENT_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of failed capture statements.</td>
</tr>
<tr>
<td>CAPTURE_FAILED_FETCH_COUNT</td>
<td>BIGINT</td>
<td>Displays the number of failed fetch operations during capture.</td>
</tr>
<tr>
<td>CAPTURE_SIZE</td>
<td>BIGINT</td>
<td>Displays the size of captured workload file in bytes.</td>
</tr>
<tr>
<td>PREPROCESS_SIZE</td>
<td>BIGINT</td>
<td>Displays the total amounts of file sizes generated by workload preprocess in bytes.</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>Displays the error code generated by system during capturing.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>NVARCHAR (5000)</td>
<td>Displays the error message generated by system during capturing.</td>
</tr>
</tbody>
</table>

**Related Information**

- CREATE WORKLOAD CLASS Statement (Workload Management) [page 929]
- ALTER WORKLOAD CLASS Statement (Workload Management) [page 678]
- ALTER WORKLOAD MAPPING Statement (Workload Management) [page 681]
- CREATE WORKLOAD MAPPING Statement (Workload Management) [page 934]
- DROP WORKLOAD CLASS Statement (Workload Management) [page 989]
- DROP WORKLOAD MAPPING Statement (Workload Management) [page 990]

**Workload Management**

**6.2.374 M_XS_APPLICATIONS System View**

Provides information about applications deployed and running in SAP HANA XS.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
</tbody>
</table>
### Related Information

- M_XS_APPLICATION_ISSUES System View [page 2293]
- M_XS_PUBLIC_URLS System View [page 2294]
- M_XS_SESSIONS System View [page 2295]
- ALL_AUDIT_LOG System View [page 1490]
- AUDIT_LOG System View [page 1501]
- Database Role Details
- Privileges

### 6.2.375 M_XS_APPLICATION_ISSUES System View

Lists issues related to applications in SAP HANA XS.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the name of the application.</td>
</tr>
<tr>
<td>ERROR_MESSAGE</td>
<td>VARCHAR(1024)</td>
<td>Displays the textual description of the issue.</td>
</tr>
</tbody>
</table>
Related Information

M_XS_APPLICATIONS System View [page 2292]
ALL_AUDIT_LOG System View [page 1490]
AUDIT_LOG System View [page 1501]
Database Role Details
Privileges

6.2.376 M_XS_PUBLIC_URLS System View

Provides a list of URLs under which this database can be reached.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP_URL</td>
<td>VARCHAR(270)</td>
<td>Displays the public HTTP URL of the XS Engine.</td>
</tr>
<tr>
<td>HTTPS_URL</td>
<td>VARCHAR(270)</td>
<td>Displays the public HTTPS URL of the XS Engine.</td>
</tr>
<tr>
<td>PURPOSE</td>
<td>VARCHAR(32)</td>
<td>Displays the purpose of these URLs.</td>
</tr>
</tbody>
</table>

Related Information

M_XS_SESSIONS System View [page 2295]
M_XS_APPLICATIONS System View [page 2292]
M_XS_APPLICATION_ISSUES System View [page 2293]
6.2.377 M_XS_SESSIONS System View

Provides information about sessions running in SAP HANA XS.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Displays the host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Displays the internal port.</td>
</tr>
<tr>
<td>XS_SESSION_ID</td>
<td>BIGINT</td>
<td>Displays the identifier of XS session running on this host.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td>Displays the user name.</td>
</tr>
<tr>
<td>REQUEST_COUNT</td>
<td>INTEGER</td>
<td>Displays the number of HTTP requests that invoked this XS session.</td>
</tr>
<tr>
<td>AUTHENTICATION_METHOD</td>
<td>VARCHAR(48)</td>
<td>Displays the method used for authenticating this XS session.</td>
</tr>
<tr>
<td>IS_DEBUGGED</td>
<td>VARCHAR(5)</td>
<td>Displays whether a debugger is currently attached to this XS session: TRUE/FALSE.</td>
</tr>
<tr>
<td>REFERENCE_COUNT</td>
<td>INTEGER</td>
<td>Displays the current number of references to this XS session.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the point in time when this XS session started.</td>
</tr>
<tr>
<td>LAST_REQUEST_TIME</td>
<td>TIMESTAMP</td>
<td>Displays the point in time of last activity in the XS session.</td>
</tr>
</tbody>
</table>

Related Information

SET [SESSION] Statement (Session Management) [page 1171]
UNSET [SESSION] Statement (Session Management) [page 1183]
Session-Specific Information for Connections
7 System Views Reference for Additional SAP HANA Contexts

This section contains system view information that can be used in other SAP HANA contexts.

SAP HANA server software and tools are used in many SAP HANA platform and installation scenarios. The feature capability of the SAP HANA server can depend on the type of SAP HANA license as well as any additional capabilities that have been installed separately. Refer to the Feature Scope Description for SAP HANA for your specific SAP HANA version located on the SAP HANA Platform webpage for information about the capabilities available for your license and installation scenario.

⚠️ Caution

This guide contains syntax variations for different product contexts. These are handled as separate reference topics that have titles that announce the context in square brackets at the end of the title. The guide is also organized to keep reference topics together for a context. Always be sure you are using the SQL reference topic specific to your context.

7.1 Dynamic Tiering

⚠️ Caution

SAP HANA dynamic tiering is deprecated. Consider migrating to SAP HANA Native Storage Extension (NSE).

⚠️ Caution

This guide contains syntax variations for different product contexts. These are handled as separate reference topics that have titles that announce the context in square brackets at the end of the title. The guide is also organized to keep reference topics together for a context. Always be sure you are using the SQL reference topic specific to your context.

- M_ES_CONNECTIONS System View [Dynamic Tiering] [page 2297]
  Processes running in SAP HANA dynamic tiering.
- M_ES,DBSPACE_FILES System View [Dynamic Tiering] [page 2298]
  Shows information of all dbspace files in the extended storage.
- M_ES,DBSPACES System View [Dynamic Tiering] [page 2299]
  Shows the dbspace configured in extended storage.
- M_ESDELTA_MEMORY System View [Dynamic Tiering] [page 2300]
  Delta memory utilization in SAP HANA dynamic tiering.
- M_ESDELTA,MERGE_STATISTICS System View [Dynamic Tiering] [page 2301]
History of internal and user driven merge operations for delta enabled extended tables.

**M_ES_EXPENSIVE_STATEMENTS System View [Dynamic Tiering] [page 2302]**
Provides an interface to give system administrators more visibility into where time and resources are being spent during statement executions that involve extended storage. The information is a result of the union of M_EXPENSIVE_STATEMENTS and MREMOTE_STATEMENTS.

**M_ES_LOCKS System View [Dynamic Tiering] [page 2302]**
Locks held in SAP HANA dynamic tiering.

**M_ES_OVERVIEW System View [Dynamic Tiering] [page 2303]**
Status information of SAP HANA dynamic tiering as named value pair.

**M_ES_RESULT_CACHE [Dynamic Tiering] [page 2304]**
Status information on the result cache storage for SAP HANA dynamic tiering.

**M_ES_TABLES System View [Dynamic Tiering] [page 2305]**
Shows information for all extended tables in extended storage.

**M_ES_TRANSACTIONS System View [Dynamic Tiering] [page 2306]**
Active transactions in SAP HANA dynamic tiering.

**Additional System Views [Dynamic Tiering] [page 2308]**
Views including information pertaining to, but not limited to, SAP HANA dynamic tiering.

### 7.1.1 M_ES_CONNECTIONS System View [Dynamic Tiering]
Processes running in SAP HANA dynamic tiering.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONNECTION_ID</td>
<td>BIGINT</td>
<td>Connection ID. This connection ID can join the REMOTE_CONNECTION_ID of MREMOTE_STATEMENTS.</td>
</tr>
<tr>
<td>INTERNAL_CONNECTION_ID</td>
<td>BIGINT</td>
<td>Internal Connection ID.</td>
</tr>
<tr>
<td>USER_ID</td>
<td>VARCHAR(255)</td>
<td>User ID for the connection.</td>
</tr>
<tr>
<td>REQUEST_TYPE</td>
<td>VARCHAR(255)</td>
<td>Type of the last request.</td>
</tr>
<tr>
<td>COMMAND_TYPE</td>
<td>VARCHAR(32)</td>
<td>Specific type of the command the command is executing.</td>
</tr>
<tr>
<td>LOCK_OWNER_CONNECTION_ID</td>
<td>BIGINT</td>
<td>The connection ID on which this connection is blocked; 0 if not blocked on any connection.</td>
</tr>
<tr>
<td>LOCK_OWNER_USER_ID</td>
<td>VARCHAR(255)</td>
<td>The owner of the blocking connection; NULL if there is no blocking connection.</td>
</tr>
<tr>
<td>CURSOR_COUNT</td>
<td>INTEGER</td>
<td>The number of active cursors in the connection; 0 if no cursors.</td>
</tr>
</tbody>
</table>
### Column Descriptions

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>THREAD_COUNT</td>
<td>INTEGER</td>
<td>The number of threads with the connection. At least one thread is started as soon as the connection is opened, so the minimum value is 1.</td>
</tr>
<tr>
<td>IDLE_DURATION</td>
<td>INTEGER</td>
<td>The time in seconds since the last SA request was issued through the connection; in case of no last SA command, the time since '01-01-2000' is displayed.</td>
</tr>
<tr>
<td>INTERNAL_IDLE_DURATI</td>
<td>INTEGER</td>
<td>The time in seconds since the last command was issued through the connection; in case of no command, the time since '01-01-2000' is displayed.</td>
</tr>
<tr>
<td>TEMP_TABLE_SIZE</td>
<td>BIGINT</td>
<td>The size of temporary table space in MBs; 0 if no temporary table space is used.</td>
</tr>
<tr>
<td>TEMP_WORKSPACE_SIZE</td>
<td>BIGINT</td>
<td>The size of temporary workspace in MBs; 0 if no temporary workspace is used.</td>
</tr>
<tr>
<td>LAST_REQUEST_TIME</td>
<td>TIMESTAMP</td>
<td>The time at which the last request for this connection started.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>The time the connection was created.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>BIGINT</td>
<td>The transaction ID of the current transaction on the connection. This is the same as the transaction ID in the esserver.trc file by the BeginTxn, CmtTxn, and PostCmtTxn messages, as well as the Txn ID Seq logged when the database is opened.</td>
</tr>
</tbody>
</table>

### M_ES_DBSPACE_FILES System View [Dynamic Tiering]

Shows information of all dbspace files in the extended storage.

This monitoring view provides information for the extended storage corresponding to the connected HANA database.

Only users with CATALOG READ, DATA ADMIN, or EXTENDED STORAGE ADMIN system privilege can view the content of the M_ES_DBSPACE_FILES view. The view is empty for all other database users.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Volume ID.</td>
</tr>
<tr>
<td>DBSPACE_NAME</td>
<td>VARCHAR(128)</td>
<td>Name of the dbspace using this file.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>DBSPACE_FILE_NAME</td>
<td>VARCHAR(128)</td>
<td>Name of the dbspace file.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(7)</td>
<td>Status of the file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• OFFLINE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ONLINE</td>
</tr>
<tr>
<td>MODE</td>
<td>VARCHAR(9)</td>
<td>Mode of the file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• READONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• READWRITE</td>
</tr>
<tr>
<td>USAGE</td>
<td>DOUBLE</td>
<td>Percent of file in use by the dbspace.</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR(512)</td>
<td>Physical path of the file.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Total size of the file specified in bytes.</td>
</tr>
<tr>
<td>RESERVED_SIZE</td>
<td>BIGINT</td>
<td>Reserved space in bytes that can be added to the file.</td>
</tr>
<tr>
<td>DISK_STRIPE_SIZE</td>
<td>BIGINT</td>
<td>Disk stripe size specified in bytes.</td>
</tr>
<tr>
<td>IS_DROP_ALLOWED</td>
<td>VARCHAR(5)</td>
<td>TRUE if file can be dropped, else FALSE.</td>
</tr>
</tbody>
</table>

7.1.3 M_ES_DBSPACES System View [Dynamic Tiering]

Shows the dbspace configured in extended storage.

System and user data for the extended storage is managed in dbspaces, which can contain one or more files. This monitoring view provides information for the extended storage corresponding to the connected HANA database.

Only users with CATALOG READ, DATA ADMIN, or EXTENDED STORAGE ADMIN system privilege can view the content of the M_ES_DBSPACES view. The view is empty for all other database users.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td>Host name.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td>Internal port.</td>
</tr>
<tr>
<td>VOLUME_ID</td>
<td>INTEGER</td>
<td>Volume ID.</td>
</tr>
<tr>
<td>DBSPACE_NAME</td>
<td>VARCHAR(128)</td>
<td>Name of the dbspace.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>TYPE</td>
<td>VARCHAR(11)</td>
<td>Type of the dbspace:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MAIN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TEMPORARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SHARED_TEMP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RLV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CACHE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• CATALOG</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(7)</td>
<td>Status of the dbspace:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• OFFLINE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ONLINE</td>
</tr>
<tr>
<td>MODE</td>
<td>VARCHAR(9)</td>
<td>Mode of the dbspace:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• READONLY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• READWRITE</td>
</tr>
<tr>
<td>USAGE</td>
<td>DOUBLE</td>
<td>Percent of dbspace in use.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>BIGINT</td>
<td>Total size of the dbspace in bytes.</td>
</tr>
<tr>
<td>RESERVED_SIZE</td>
<td>BIGINT</td>
<td>Total reserved space in bytes that can be added to dbfiles in the dbspace.</td>
</tr>
<tr>
<td>FILE_COUNT</td>
<td>INTEGER</td>
<td>Total number of files in the dbspace.</td>
</tr>
<tr>
<td>READWRITE_FILE_COUNT</td>
<td>INTEGER</td>
<td>Total number of read-write files in the dbspace.</td>
</tr>
<tr>
<td>IS_DISK_STRIPING_ENABLED</td>
<td>VARCHAR(5)</td>
<td>TRUE if disk striping is enabled for the dbspace, else FALSE.</td>
</tr>
<tr>
<td>DISK_STRIP_SIZE</td>
<td>BIGINT</td>
<td>Disk stripe size in byte.</td>
</tr>
</tbody>
</table>

### 7.1.4 M_ES_DELTA_MEMORY System View [Dynamic Tiering]

Delta memory utilization in SAP HANA dynamic tiering.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Table name.</td>
</tr>
<tr>
<td>FRAGMENT_COUNT</td>
<td>INTEGER</td>
<td>Number of store fragments for this table.</td>
</tr>
<tr>
<td>TOTAL_SIZE</td>
<td>INTEGER</td>
<td>Total delta memory used in MB by this table.</td>
</tr>
<tr>
<td>DATA_SIZE</td>
<td>INTEGER</td>
<td>Delta memory used in MB to store the table data.</td>
</tr>
</tbody>
</table>
### M_ES_DELTA_MERGE_STATISTICS System View

[Dynamic Tiering]

History of internal and user driven merge operations for delta enabled extended tables.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(255)</td>
<td>Schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(255)</td>
<td>Table name.</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Time that the delta merge started.</td>
</tr>
<tr>
<td>END_TIME</td>
<td>TIMESTAMP</td>
<td>Time that the delta merge ended.</td>
</tr>
<tr>
<td>STATUS</td>
<td>VARCHAR(16)</td>
<td>Status of the merge operation: 'STARTED', 'COMPLETED', 'FAILED'</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>INTEGER</td>
<td>SQL code of the merge after completion.</td>
</tr>
<tr>
<td>ERROR_TEXT</td>
<td>NVARCHAR(255)</td>
<td>Additional error information.</td>
</tr>
<tr>
<td>MOTIVATION</td>
<td>VARCHAR(16)</td>
<td>Cause of the merger trigger: 'DML', 'DDL', 'SHUTDOWN', 'USER'</td>
</tr>
<tr>
<td>IS_BLOCKING</td>
<td>VARCHAR(5)</td>
<td>Specify if the merge operation was blocking: 'TRUE', 'FALSE'</td>
</tr>
<tr>
<td>INSERT_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Number of rows that were inserted as a result of the merge.</td>
</tr>
<tr>
<td>UPDATE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Number of rows that were inserted as a result of the merge.</td>
</tr>
<tr>
<td>DELETE_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Number of rows that were inserted as a result of the merge.</td>
</tr>
<tr>
<td>UNCOMMITTED_RECORD_COUNT</td>
<td>BIGINT</td>
<td>Number of rows that were inserted as a result of the merge.</td>
</tr>
</tbody>
</table>
7.1.6 M_ES_EXPENSIVE_STATEMENTS System View [Dynamic Tiering]

Provides an interface to give system administrators more visibility into where time and resources are being spent during statement executions that involve extended storage. The information is a result of the union of M_EXPENSIVE_STATEMENTS and M_REMOTE_STATEMENTS.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONNECTION_ID</td>
<td>INTEGER</td>
<td>Connection ID.</td>
</tr>
<tr>
<td>TRANSACTION_ID</td>
<td>INTEGER</td>
<td>Transaction ID.</td>
</tr>
<tr>
<td>STATEMENT_ID</td>
<td>VARCHAR(20)</td>
<td>Statement ID.</td>
</tr>
<tr>
<td>OPERATION</td>
<td>NVARCHAR(5000)</td>
<td>Type of operation (such as prepare, execute, fetch, close).</td>
</tr>
<tr>
<td>START_TIME</td>
<td>TIMESTAMP</td>
<td>Statement start time.</td>
</tr>
<tr>
<td>DURATION_MICROSEC</td>
<td>BIGINT</td>
<td>Statement duration.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>Statement string.</td>
</tr>
<tr>
<td>RECORDS</td>
<td>BIGINT</td>
<td>Number of records.</td>
</tr>
</tbody>
</table>

7.1.7 M_ES_LOCKS System View [Dynamic Tiering]

Locks held in SAP HANA dynamic tiering.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONNECTION_NAME</td>
<td>VARCHAR(255)</td>
<td>The name of the current connection.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>BIGINT</td>
<td>Connection id that has the lock. This connection id can join the REMOTE_CONNECTION_ID of M_REMOTE_STATEMENTS.</td>
</tr>
<tr>
<td>USER_ID</td>
<td>VARCHAR(255)</td>
<td>User id associated with this connection.</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>VARCHAR(32)</td>
<td>The type of table. This type is either 'BASE TABLE' for a table, 'GLOBAL TEMPORARY TABLE' for global temporary table, or 'MATERIALIZED VIEW' for a materialized view.</td>
</tr>
</tbody>
</table>
### Column name | Data type | Description
--- | --- | ---
SCHEMA_NAME | NVARCHAR(256) | The schema name of the table.
TABLE_NAME | NVARCHAR(256) | Table on which the lock is held.
INDEX_ID | BIGINT | The index ID or NULL.
LOCK_CLASS | VARCHAR(8) | The lock class. One of SCHEMA, ROW, TABLE, or POSITION.
LOCK_DURATION_TYPE | VARCHAR(16) | The duration of the lock. One of TRANSACTION, POSITION, or CONNECTION.
LOCK_TYPE | VARCHAR(10) | Depending on the lock class, it could be one of:
  - SHARED, EXCLUSIVE for SCHEMA
  - READ, INTENT, READPK, WRITE, WRITE-NOPK, SURROGATE for ROW
  - SHARED, INTENT, EXCLUSIVE for TABLE
  - PHANTOM, INSERT for POSITION
ROW_IDENTIFIER | BIGINT | The identifier for the row the lock starts on, or NULL.
ROW_RANGE | BIGINT | The number of contiguous rows that are locked. Row locks in the DELTA store can either be a single row, or a range of rows.

### 7.1.8 M_ES_OVERVIEW System View [Dynamic Tiering]

Status information of SAP HANA dynamic tiering as named value pair.

### Structure

### Column name | Data type | Description
--- | --- | ---
CATEGORY | VARCHAR(32) | Describes the main category type for which the parameters and values are shown. The following categories are reported:
  - GENERAL
  - BUFFERS
  - MEMORY
  - STORAGE
  - DELTA
  - BACKUP
  - VERSIONING
7.1.9 M_ES_RESULT_CACHE [Dynamic Tiering]

Status information on the result cache storage for SAP HANA dynamic tiering.

Only users with EXTENDED STORAGE ADMIN system privilege can view the content of the M_ES_RESULT_CACHE view. The view is empty for all other database users.

There is one row for each result in the result cache.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>The schema name.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>The name of the table.</td>
</tr>
<tr>
<td>COLUMNS</td>
<td>NCLOB</td>
<td>The columns included in the result as a comma separated list.</td>
</tr>
<tr>
<td>PREDICATES</td>
<td>NCLOB</td>
<td>The predicates used to generate the result as a comma separated list.</td>
</tr>
<tr>
<td>COMMIT_ID</td>
<td>BIGINT</td>
<td>The version number of the table.</td>
</tr>
<tr>
<td>STATEMENT_STRING</td>
<td>NCLOB</td>
<td>The query that generated the result.</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>The timestamp the result cache entry is generated and stored.</td>
</tr>
<tr>
<td>CREATE_DURATION</td>
<td>BIGINT</td>
<td>The number of milliseconds it took to generate the original result.</td>
</tr>
<tr>
<td>CREATE_CONNECTION_ID</td>
<td>BIGINT</td>
<td>The connection ID that generated the result. This connection id can join the REMOTE_CONNECTION_ID of M_REMOTE_STATEMENTS.</td>
</tr>
<tr>
<td>LAST_ACCESS_TIME</td>
<td>TIMESTAMP</td>
<td>The last time the result cache entry was accessed by a user.</td>
</tr>
<tr>
<td>LAST_CONNECTION_ID</td>
<td>BIGINT</td>
<td>The last connection ID that accessed the result. This connection id can join the REMOTE_CONNECTION_ID of M_REMOTE_STATEMENTS.</td>
</tr>
</tbody>
</table>

NAME
VARCHAR(255)
Name of the parameter of which the information is provided in value column.

VALUE
VARCHAR(255)
Value of the parameter.
<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIT_COUNT</td>
<td>BIGINT</td>
<td>The number of accesses on the cache entry.</td>
</tr>
<tr>
<td>SUBSET_HIT_COUNT</td>
<td>BIGINT</td>
<td>The number of accesses on the cache entry that only used a subset of the result.</td>
</tr>
<tr>
<td>EXACT_MATCH_HIT_COUNT</td>
<td>BIGINT</td>
<td>The number of accesses on the cache entry that used the whole result.</td>
</tr>
<tr>
<td>RESULT_SIZE</td>
<td>BIGINT</td>
<td>The size of memory occupied by the result cache entry (in bytes).</td>
</tr>
</tbody>
</table>

### 7.1.10 M_ES_TABLES System View [Dynamic Tiering]

Shows information for all extended tables in extended storage.

An extended table is a table created in extended storage. This monitoring view provides information for the extended storage corresponding to the connected HANA database.

Users with DATA ADMIN system privilege can view information about all extended tables in the M_ES_TABLES view. Users without this system privilege can see information only for extended tables owned by that user or for extended tables to which they have SELECT permission.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td>Schema name for extended table.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>NVARCHAR(256)</td>
<td>Table name for extended table.</td>
</tr>
<tr>
<td>PART_ID</td>
<td>INTEGER</td>
<td>Logical partition ID. Value is 0 for extended tables.</td>
</tr>
<tr>
<td>IS_DELTA_ENABLED</td>
<td>VARCHAR(5)</td>
<td>TRUE if it is a RLV table, otherwise FALSE.</td>
</tr>
<tr>
<td>TABLE_SIZE</td>
<td>BIGINT</td>
<td>Size of compressed data in bytes.</td>
</tr>
<tr>
<td>RECORD_COUNT</td>
<td>BIGINT</td>
<td>Row count in extended table.</td>
</tr>
<tr>
<td>LAST_MODIFIED_TIME</td>
<td>TIMESTAMP</td>
<td>Timestamp when table was last modified.</td>
</tr>
</tbody>
</table>
7.1.11 M_ES_TRANSACTIONS System View [Dynamic Tiering]

Active transactions in SAP HANA dynamic tiering.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSACTION_ID</td>
<td>BIGINT</td>
<td>The transaction ID of this transaction. It is assigned during begin transaction. It appears in the esserver.trc file by the BeginTxn, CmtTxn, and PostCmtTxn messages, and is the same as the Txn ID Seq that is logged when the database is opened.</td>
</tr>
<tr>
<td>CONNECTION_ID</td>
<td>BIGINT</td>
<td>Connection Id. This connection Id can join with REMOTE_CONNECTION_ID of M_REMOTE_STATEMENTS.</td>
</tr>
<tr>
<td>INTERNAL_CONNECTION_ID</td>
<td>BIGINT</td>
<td>The 10-digit connection ID that is included as part of all messages in the esserver.trc file. This is a monotonically increasing integer unique within a server session.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>VARCHAR(255)</td>
<td>The name of the application.</td>
</tr>
<tr>
<td>USER_ID</td>
<td>VARCHAR(255)</td>
<td>The user ID for the connection.</td>
</tr>
<tr>
<td>COMMIT_ID</td>
<td>BIGINT</td>
<td>The ID assigned by the transaction manager when the transaction commits. For active transactions, the COMMIT_ID is 0 (zero).</td>
</tr>
<tr>
<td>VERSION_ID</td>
<td>BIGINT</td>
<td>A value of 0 indicates that the transaction is unversioned, and the VERSION_ID has not been assigned.</td>
</tr>
<tr>
<td>TRANSACTION_STATE</td>
<td>VARCHAR(16)</td>
<td>The state of the transaction. Transaction states are: NONE, ACTIVE, ROLLING_BACK, ROLLED_BACK, COMMITTING, COMMITTED, and APPLIED. NONE, ROLLING_BACK, ROLLED_BACK, COMMITTING and APPLIED are transient states with a very small life span.</td>
</tr>
<tr>
<td>MAIN_TABLE_USED_SIZE</td>
<td>BIGINT</td>
<td>The number of MBs of ES_USER space created by this transaction.</td>
</tr>
<tr>
<td>MAIN_TABLE_FREED_SIZE</td>
<td>BIGINT</td>
<td>The number of MBs of ES_USER space dropped by this transaction, but which persist on disk in the store because the space is visible in other database versions or other savepoints of this transaction.</td>
</tr>
<tr>
<td>TEMP_TABLE_USED_SIZE</td>
<td>BIGINT</td>
<td>The number of MBs of ES_TEMP space created by this transaction for storage of DT temporary table data.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TEMP_TABLE_FREED_SIZE</td>
<td>BIGINT</td>
<td>The number of MBs of ESP_TEMP space dropped by this transaction, but which persist on disk in the DT temporary store because the space is visible to DT cursors or is owned by other savepoints of this transaction.</td>
</tr>
<tr>
<td>TEMP_WORKSPACE_SIZE</td>
<td>BIGINT</td>
<td>For ACTIVE transactions, a snapshot of the work space in MBs in use at this instant by this transaction, such as sorts, hashes, and temporary bitmaps. This number changes depending on the usage as memory is allocated and freed. When the transaction is no longer active, this column is zero.</td>
</tr>
<tr>
<td>TRANSACTION_CREATE_TIME</td>
<td>TIMESTAMP</td>
<td>The time the transaction began.</td>
</tr>
<tr>
<td>CURSOR_COUNT</td>
<td>INTEGER</td>
<td>The number of open DT cursors that reference this transaction control block. If the transaction is ACTIVE, it indicates the number of open cursors created within the transaction. If the transaction is COMMITTED, it indicates the number of hold cursors that reference a database version owned by this transaction.</td>
</tr>
<tr>
<td>INTERNAL_SAVEPOINT_COUNT</td>
<td>INTEGER</td>
<td>The number of savepoint structures that exist within the transaction control block. Savepoints may be created and released implicitly. Therefore, this number does not indicate the number of user-created savepoints within the transaction.</td>
</tr>
<tr>
<td>INTERNAL_SAVEPOINT_NUMBER</td>
<td>INTEGER</td>
<td>The active savepoint number of the transaction. This is an implementation detail and might not reflect a user-created savepoint.</td>
</tr>
<tr>
<td>GLOBAL_TRANSACTION_ID</td>
<td>BIG_INT</td>
<td>The global transaction id associated with the current transaction, 0 (zero) if none.</td>
</tr>
<tr>
<td>VERSIONING_TYPE</td>
<td>VARCHAR(32)</td>
<td>The snapshot versioning type of the transaction; either table-level (the default), or row-level. Row-level snapshot versioning (DELTA) applies only to DELTA-enabled tables. Once a transaction is started, this value cannot change.</td>
</tr>
<tr>
<td>IS_BLOCKED</td>
<td>VARCHAR(5)</td>
<td>Indicates if connection blocking is enabled (TRUE) or disabled (FALSE). You set connection blocking using the BLOCKING database option. If true, the transaction blocks, meaning it waits for a conflicting lock to release before it attempts to retry the lock request.</td>
</tr>
<tr>
<td>BLOCKING_TIME_OUT</td>
<td>INTEGER</td>
<td>Indicates the time, in milliseconds, a transaction waits for a locking conflict to clear. You set the timeout threshold using the BLOCKING_TIME_OUT database option. A value of 0 (default) indicates that the transaction waits indefinitely.</td>
</tr>
</tbody>
</table>
## 7.1.12 Additional System Views [Dynamic Tiering]

Views including information pertaining to, but not limited to, SAP HANA dynamic tiering.

<table>
<thead>
<tr>
<th>View</th>
<th>Information Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANTED_PRIVILEGES System View</td>
<td>Privileges granted to users and roles.</td>
</tr>
<tr>
<td>GRANTED_ROLES System View</td>
<td>Roles granted to users or other roles.</td>
</tr>
<tr>
<td>INDEX_COLUMNS System View</td>
<td>Columns of indexes.</td>
</tr>
<tr>
<td>INDEXES System View</td>
<td>Indexes on tables.</td>
</tr>
<tr>
<td>M_BACKUP_CATALOG System View</td>
<td>Common data for all backup catalog entries.</td>
</tr>
<tr>
<td>M_BACKUP_CATALOG_FILES System View</td>
<td>Location information of all backup catalog entries.</td>
</tr>
<tr>
<td>M_BACKUP_SIZE_ESTIMATIONS System View</td>
<td>Estimated size of the next data backup.</td>
</tr>
<tr>
<td>M_CONNECTIONS System View</td>
<td>Detailed information on connections between a client and database, including connection status, client information, connection type, and resource utilization.</td>
</tr>
<tr>
<td>M_DATA_STATISTICS System View</td>
<td>Lists data statistics generated when you query column and row store object.</td>
</tr>
<tr>
<td>M_SYSTEM_DATA_STATISTICS System View</td>
<td>Lists data statistics generated by the server when you query a column and row store object.</td>
</tr>
<tr>
<td>M_DISKS System View</td>
<td>Disk configuration and utilization of the host machine.</td>
</tr>
<tr>
<td>M_EXPORT_BINARY_STATUS System View</td>
<td>Export status information for the current session.</td>
</tr>
<tr>
<td>M_FEATURES System View</td>
<td>Information about all supported features.</td>
</tr>
<tr>
<td>M_HISTORY_INDEX_LAST_COMMIT_ID System View</td>
<td>The last commit ID of a history index for each session.</td>
</tr>
<tr>
<td>M_HOST_INFORMATION System View</td>
<td>Host information, such as machine and OS configuration.</td>
</tr>
<tr>
<td>M_HOST_RESOURCE_UTILIZATION System View</td>
<td>Host resource utilization. CPU time is in milliseconds and added across all cores since system start.</td>
</tr>
<tr>
<td>M_IMPORT_BINARY_STATUS System View</td>
<td>Import status information for the current session.</td>
</tr>
<tr>
<td>M_INIFILE_CONTENTS System View</td>
<td>Configuration information from INI files.</td>
</tr>
<tr>
<td>M_INIFILES System View</td>
<td>All configuration files.</td>
</tr>
<tr>
<td>M_LANDSCAPE_HOST_CONFIGURATION System View</td>
<td>Host roles in a distributed landscape.</td>
</tr>
<tr>
<td>M_MERGED_TRACES System View</td>
<td>The merged content of the server trace files for all of the SAP HANA services.</td>
</tr>
<tr>
<td>M_REMOTE_STATEMENTS System View</td>
<td>Detailed information on executed remote queries. Information includes query status, number of fetched rows.</td>
</tr>
<tr>
<td>M_SERVICE_MEMORY System View</td>
<td>Detailed information on memory utilization by services.</td>
</tr>
<tr>
<td>M_SERVICE_STATISTICS System View</td>
<td>Statistics on active services.</td>
</tr>
<tr>
<td>M_SERVICE_TYPES System View</td>
<td>Service types.</td>
</tr>
<tr>
<td>M_SERVICES System View</td>
<td>The status of all services.</td>
</tr>
<tr>
<td>View</td>
<td>Information Displayed</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>M_SYSTEM_LIMITS System View  [page 2192]</td>
<td>Information on system limits.</td>
</tr>
<tr>
<td>M_TABLE_PARTITIONS System View  [page 2209]</td>
<td>Provides information regarding partition-specific memory and disk usage for partitioned tables. The DISK_SIZE column displays the size on disk for all extended storage partitions.</td>
</tr>
<tr>
<td>M_TABLE_PRUNING_STATISTICS System View  [page 2218]</td>
<td>Provides an interface to access statistics for historical data.</td>
</tr>
<tr>
<td>M_TABLES System View  [page 2200]</td>
<td>Information on row and column tables.</td>
</tr>
<tr>
<td>M_TRACEFILE_CONTENTS System View  [page 2247]</td>
<td>SAP HANA information from trace files.</td>
</tr>
<tr>
<td>M_TRACEFILES System View  [page 2246]</td>
<td>All trace files.</td>
</tr>
<tr>
<td>M_VOLUME_FILES System View  [page 2257]</td>
<td>Information about volume files.</td>
</tr>
<tr>
<td>M_VOLUME_SIZES System View  [page 2282]</td>
<td>Volume sizes used by SAP HANA servers.</td>
</tr>
<tr>
<td>M_VOLUMES System View  [page 2256]</td>
<td>Volumes used by SAP HANA servers.</td>
</tr>
<tr>
<td>TRANSACTION_HISTORY System View  [page 1693]</td>
<td>Committed transactions and their users.</td>
</tr>
<tr>
<td>USERS System View  [page 1698]</td>
<td>All users.</td>
</tr>
</tbody>
</table>

### 7.2 Streaming Analytics

⚠es Caution

This guide contains syntax variations for different product contexts. These are handled as separate reference topics that have titles that announce the context in square brackets at the end of the title. The guide is also organized to keep reference topics together for a context. Always be sure you are using the SQL reference topic specific to your context.

- **M_STREAMING_APPLICATIONS System View [Streaming Analytics] [page 2310]**
  Provides a list of streaming analytics applications running in the SAP HANA streaming analytics cluster.

- **M_STREAMING_PROJECTS System View [Streaming Analytics] [page 2311]**
  Provides information and statistics about each streaming analytics project deployed in the SAP HANA streaming analytics cluster.

- **M_STREAMING_PROJECT_ADAPTERS System View [Streaming Analytics] [page 2314]**
  Contains information about internal adapters attached to each SAP HANA streaming analytics project.

- **M_STREAMING_PROJECT_ADAPTER_STATISTICS System View [Streaming Analytics] [page 2316]**
  Contains custom statistics for both external and internal adapters.

- **M_STREAMING_PROJECT_GD_SUBSCRIPTIONS System View [Streaming Analytics] [page 2317]**
  Contains information about active and inactive GD (Guaranteed Delivery) Subscriptions. The subscribers can be either external subscribers or internal adapters that support GD.
M_STREAMING_PROJECT_PUBLISHERS System View [Streaming Analytics] [page 2318]
Contains information and statistics about streaming publishers for each SAP HANA streaming analytics project, including external adapters.

M_STREAMING_PROJECT_SUBSCRIBERS System View [Streaming Analytics] [page 2321]
Contains information and statistics about streaming subscribers for each SAP HANA streaming analytics project, including external adapters.

M_STREAMING_PROJECT_STREAMS System View [Streaming Analytics] [page 2324]
Provides information and statistics about each stream in an SAP HANA streaming analytics project.

M_STREAMING_SCHEMAS System View [Streaming Analytics] [page 2326]
Provides a list of the streaming schemas in the SAP HANA streaming analytics cluster.

M_STREAMING_SERVICES System View [Streaming Analytics] [page 2327]
Provides a list of logical node names in the SAP HANA streaming analytics cluster.

Additional Views For Streaming Analytics [page 2327]
Views including information pertaining to, but not limited to, SAP HANA streaming analytics.

7.2.1 M_STREAMING_APPLICATIONS System View [Streaming Analytics]

Provides a list of streaming analytics applications running in the SAP HANA streaming analytics cluster.

Only users with a MONITOR role can view the content of this system view. The view is empty for all other users.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The schema in which the application is deployed.</td>
</tr>
<tr>
<td>APPLICATION_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name of the streaming analytics application.</td>
</tr>
<tr>
<td>APPLICATION_TYPE</td>
<td>VARCHAR(32)</td>
<td></td>
<td>Application type. For example PROJECT, HA_PROJECT, TOOLKIT_ADAPTER</td>
</tr>
<tr>
<td>APPLICATION_INSTANCE</td>
<td>INTEGER</td>
<td></td>
<td>The instance number of the streaming analytics application. A schema may have multiple instances of an application running. As, for example, in HA or distributed processing.</td>
</tr>
<tr>
<td>APPLICATION_INSTANCE_COUNT</td>
<td>INTEGER</td>
<td></td>
<td>The number of instances that are running or will be started.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SERVICE_ALIAS</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Name of the service on which the application is running.</td>
</tr>
<tr>
<td>REQUESTED_STATUS</td>
<td>VARCHAR(32)</td>
<td></td>
<td>The status the application should be in, as requested by the client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STOPPED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STARTED_RUNNING</td>
</tr>
<tr>
<td>CURRENT_STATUS</td>
<td>VARCHAR(32)</td>
<td></td>
<td>The status the application is currently in.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• START</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STARTING</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STARTED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STARTED_RUNNING</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STOPPING</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STOPPED_FALSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STOPPED_FALSE_START</td>
</tr>
<tr>
<td>IS_FAILOVER_ENABLED</td>
<td>VARCHAR(5)</td>
<td></td>
<td>Indicates whether the project should be started on another node if the project or the node fails.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• TRUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• FALSE</td>
</tr>
<tr>
<td>FAILURE_COUNT</td>
<td>INTEGER</td>
<td></td>
<td>The number of failures that have occurred during the failure interval.</td>
</tr>
<tr>
<td>FAILURE_INTERVAL</td>
<td>INTEGER</td>
<td></td>
<td>The interval, in seconds, during which, if the failure count reaches the maximum failures per interval, the application is shut down</td>
</tr>
<tr>
<td>MAX_FAILURES_PER_INTERVAL</td>
<td>INTEGER</td>
<td></td>
<td>The number of failures that can occur in the specified failure interval before the application is shut down</td>
</tr>
</tbody>
</table>

### 7.2.2 M_STREAMING_PROJECTS System View [Streaming Analytics]

Provides information and statistics about each streaming analytics project deployed in the SAP HANA streaming analytics cluster.

Only users with a MONITOR role can view the content of this system view. The view is empty for all other users.
### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Schema in which the project is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Name given to the project. It corresponds to the APPLICATION_NAME in the M_STREAMING_APPLICATIONS system view for APPLICATION_TYPE “PROJECT” or “HA_PROJECT”.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td></td>
<td>Instance number of the project. It corresponds to the APPLICATION_INSTANCE in the M_STREAMING_APPLICATIONS system view for APPLICATION_TYPE “PROJECT” or “HA_PROJECT”.</td>
</tr>
<tr>
<td>CPU_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>Total CPU usage, as a percentage, used by the project since the last update. Total CPU usage is equal to the sum of the system CPU usage plus the user CPU usage. Valid values range from 00 to 100.00. On multi-core machines, the percentage is relative to the total number of available cores. For example, a value of 100.00 indicates a usage of 100% on all cores on the machine.</td>
</tr>
<tr>
<td>CPU_SYSTEM_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>System CPU usage (Kernel usage on Windows), as a percentage, used by the project since the last update. Valid values range from 0.0 to 100.00. On multi-core machines, the percentage is relative to the total number of available cores. For example, a value of 100.00 indicates a usage of 100% of all cores on the machine.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CPU_USER_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>User CPU usage, as a percentage, used by the project since the last update. On multi-core machines, the percentage is relative to the total number of available cores. For example, a value of 100.00 indicates a usage of 100% on all cores on the machine.</td>
</tr>
<tr>
<td>TOTAL_CPU_TIME</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Total CPU time for the project, in microseconds. Total CPU time is equal to the sum of the system CPU time plus the user CPU time.</td>
</tr>
<tr>
<td>TOTAL_CPU_SYSTEM_TIME</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Total system CPU time for the project, in microseconds.</td>
</tr>
<tr>
<td>TOTAL_CPU_USER_TIME</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Total user CPU time for the project, in microseconds.</td>
</tr>
<tr>
<td>PROJECT_DURATION</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Duration of lapsed real time since the project was started, in microseconds.</td>
</tr>
<tr>
<td>VIRTUAL_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_BYTE</td>
<td>Total amount of virtual memory, in bytes, used by the project at the time of the update.</td>
</tr>
<tr>
<td>RESIDENT_MEMORY_SIZE</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_BYTE</td>
<td>Total amount of system memory (RSS), in bytes, used by the project at the time of the update.</td>
</tr>
<tr>
<td>THREAD_COUNT</td>
<td>INTEGER</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>Total number of threads used by the project at the time of the update.</td>
</tr>
<tr>
<td>MONITORING_INTERVAL</td>
<td>INTEGER</td>
<td>MONITOR_UNIT_SEC</td>
<td>Specifies, in seconds, how often stream and client statistics are gathered. A value of 0 indicates that no statistics are gathered for the streams and clients.</td>
</tr>
<tr>
<td>IS_ACTIVE_ACTIVE_ENABLED</td>
<td>VARCHAR(5)</td>
<td>MONITOR_UNIT_BOOL</td>
<td>Indicates whether active-active high availability mode is enabled for the project.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>ACTIVE_ACTIVE_ROLE</td>
<td>VARCHAR(16)</td>
<td></td>
<td>Indicates whether the project instance plays the primary or secondary role in active-active high availability mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• PRIMARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• SECONDARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• UNKNOWN (a transitional state)</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td></td>
<td>Most recent time that statistics were updated for the project.</td>
</tr>
</tbody>
</table>

### 7.2.3 M_STREAMING_PROJECT_ADAPTERS System View [Streaming Analytics]

Contains information about internal adapters attached to each SAP HANA streaming analytics project

Only users with a MONITOR role can view the content of this system view. The view is empty for all other users.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Schema in which the project is deployed</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Name given to the project. Corresponds to the APPLICATION_NAME in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE “PROJECT” or “HA_PROJECT”.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td></td>
<td>Instance number of the project. Corresponds to the APPLICATION_INSTANCE in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE “PROJECT” or “HA_PROJECT”.</td>
</tr>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name of the adapter, as defined in the project.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ADAPTER_TYPE</td>
<td>VARCHAR(32)</td>
<td></td>
<td>The adapter type defined in the ATTACH ADAPTER statement.</td>
</tr>
<tr>
<td>STREAM_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name of the stream on which the adapter is defined.</td>
</tr>
<tr>
<td>IS_INPUT</td>
<td>VARCHAR(5)</td>
<td>MONITOR_UNIT_BOOL</td>
<td>Indicates whether the connection is an input adapter or an output adapter. Values are TRUE for input connection and FALSE for output connection.</td>
</tr>
<tr>
<td>STARTUP_GROUP_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The start up group where this connector belongs.</td>
</tr>
<tr>
<td>ADAPTER_STATE</td>
<td>VARCHAR(32)</td>
<td></td>
<td>The state of the adapter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• READY – ready to be started</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• INITIAL – performing start-up and initialization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• CONTINUOUS – continuously receiving real-time data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• IDLE – currently not receiving data but attempting to re-connect the to the data source or link</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• DONE – no remaining input or output data; the adapter is about to exit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• DEAD – the adapter thread has exited. The adapter remains in this state until explicitly requested to restart.</td>
</tr>
<tr>
<td>TOTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The total number of data records recognized in the input data.</td>
</tr>
<tr>
<td>SUCCESSFUL_ROW_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The number of data records successfully processed.</td>
</tr>
<tr>
<td>UNSUCCESSFUL_ROW_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The number of data records that experienced errors.</td>
</tr>
<tr>
<td>MEMORY_USAGE</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_BYTE</td>
<td>The total memory used by the adapter, in bytes.</td>
</tr>
<tr>
<td>LAST_ERROR_TIME</td>
<td>TIMESTAMP</td>
<td></td>
<td>The last time that an error occurred.</td>
</tr>
</tbody>
</table>
### 7.2.4 M_STREAMING_PROJECT_ADAPTER_STATISTICS

**System View [Streaming Analytics]**

Contains custom statistics for both external and internal adapters. Only users with a MONITOR role can view the content of this system view. The view is empty for all other users.

**Structure**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The schema in which the project is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name given to the project. It corresponds to the APPLICATION_NAME in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE “PROJECT” or “HA_PROJECT”.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td></td>
<td>The instance number of the project. It corresponds to the APPLICATION_INSTANCE in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE “PROJECT” or “HA_PROJECT”.</td>
</tr>
</tbody>
</table>
# 7.2.5 M_STREAMING_PROJECT_GD_SUBSCRIPTIONS

## System View [Streaming Analytics]

Contains information about active and inactive GD (Guaranteed Delivery) Subscriptions. The subscribers can be either external subscribers or internal adapters that support GD.

Only users with a MONITOR role can view the content of this system view. The view is empty for all other users.

### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADAPTER_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Unique name of the adapter instance.</td>
</tr>
<tr>
<td>STATISTICS_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name of an adapter statistic, as defined by the adapter.</td>
</tr>
<tr>
<td>STATISTICS_VALUE</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The value of the statistic (converted to a string).</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td></td>
<td>The last time the statistic was updated for the adapter.</td>
</tr>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The schema in which the project is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name given to the project. It corresponds to the APPLICATION_NAME in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE &quot;PROJECT&quot; or &quot;HA_PROJECT&quot;.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td></td>
<td>The instance number of the project. It corresponds to the APPLICATION_INSTANCE in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE &quot;PROJECT&quot; or &quot;HA_PROJECT&quot;.</td>
</tr>
<tr>
<td>GD_KEY</td>
<td>NVARCHAR(1024)</td>
<td></td>
<td>The automatically generated key for the GD (Guaranteed Delivery) session.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------</td>
<td>------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STREAM_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name of the stream being subscribed.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name of the user doing the subscription.</td>
</tr>
<tr>
<td>GD_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name of this guaranteed delivery session.</td>
</tr>
<tr>
<td>GD_SEQUENCE_NUMBER</td>
<td>BIGINT</td>
<td></td>
<td>The sequence number of the last event committed from stream_name in this guaranteed delivery session. A value of 0 indicates that no events have been committed.</td>
</tr>
<tr>
<td>STREAMING_CONNECTION_ID</td>
<td>INTEGER</td>
<td></td>
<td>The unique handle for active external subscribers. For active internal adapters the value is always 0. Inactive subscribers have a value of -1.</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td></td>
<td>The last time statistics were updated for the GD subscription.</td>
</tr>
</tbody>
</table>

### 7.2.6 M_STREAMING_PROJECT_PUBLISHERS System View

[Streaming Analytics]

Contains information and statistics about streaming publishers for each SAP HANA streaming analytics project, including external adapters.

Only users with a MONITOR role can view the content of this system view. The view is empty for all other users.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The schema in which the project is deployed.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name given to the project. It corresponds to the APPLICATION_NAME in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE “PROJECT” or “HA_PROJECT”.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td></td>
<td>The instance number of the project. It corresponds to the APPLICATION_INSTANCE in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE “PROJECT” or “HA_PROJECT”.</td>
</tr>
<tr>
<td>STREAMING_CONNECTION_ID</td>
<td>BIGINT</td>
<td></td>
<td>A unique integer ID for the connection.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The user name provided by the client during connection establishment, displayed once the user is authenticated.</td>
</tr>
<tr>
<td>CLIENT_HOST</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The symbolic host name of the client machine, if available. If not available, host is the IP address of the client machine.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(64)</td>
<td></td>
<td>The IP address of the client machine, as a string.</td>
</tr>
<tr>
<td>CLIENT_PORT</td>
<td>INTEGER</td>
<td></td>
<td>The TCP port number from which the connection originates.</td>
</tr>
<tr>
<td>CONNECTION_TAG</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The user-set symbolic connection tag name. If not set by the user, the connection tag is NULL.</td>
</tr>
<tr>
<td>TRANSACTION_THROUGHPUT</td>
<td>REAL</td>
<td>MONITOR_UNIT_RATE</td>
<td>The client’s performance, in transactions per second, sent by the client since the last update. Envelopes and service messages count as transactions.</td>
</tr>
<tr>
<td>ROW_THROUGHPUT</td>
<td>REAL</td>
<td>MONITOR_UNIT_RATE</td>
<td>The client’s performance, in data rows per second, sent by the client since the last update.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INCREMENTAL_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The number of transactions, envelopes, and messages sent by the client since the last update.</td>
</tr>
<tr>
<td>INCREMENTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The number of data rows sent by the client since the last update.</td>
</tr>
<tr>
<td>TOTAL_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The total number of transactions, envelopes, and messages sent by the client.</td>
</tr>
<tr>
<td>TOTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The total number of data rows sent by the client.</td>
</tr>
<tr>
<td>CPU_CORE_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>Total CPU usage for the client thread, as a percentage of a single CPU core.</td>
</tr>
<tr>
<td>CPU_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>Total CPU usage for the client, since the last update. Total CPU usage is equal to the sum of the system CPU usage plus the user CPU usage.</td>
</tr>
<tr>
<td>CPU_SYSTEM_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>System CPU usage for the client, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>CPU_USER_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>User CPU usage for the client, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>TOTAL_CPU_TIME</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Total CPU time since the creation of the client, in microseconds. Total CPU time equals system CPU time plus user CPU time.</td>
</tr>
<tr>
<td>TOTAL_CPU_SYSTEM_TIME</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Total system CPU, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>TOTAL_CPU_USER_TIME</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Total user CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>CONNECT_TIME</td>
<td>TIMESTAMP</td>
<td></td>
<td>The time when the server accepts (but does not authenticate) the connection.</td>
</tr>
<tr>
<td>PUBLISHER_DURATION</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Duration of lapsed real time since the creation of the client thread.</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td></td>
<td>The most recent time that statistics were updated for the publisher.</td>
</tr>
</tbody>
</table>
7.2.7 M_STREAMING_PROJECT_SUBSCRIBERS System View

[Streaming Analytics]

Contains information and statistics about streaming subscribers for each SAP HANA streaming analytics project, including external adapters.

Only users with a MONITOR role can view the content of this system view. The view is empty for all other users.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The schema in which the project is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name given to the project. It corresponds to the APPLICATION_NAME in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE &quot;PROJECT&quot; or &quot;HA_PROJECT&quot;.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td></td>
<td>The instance number of the project. It corresponds to the APPLICATION_INSTANCE in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE &quot;PROJECT&quot; or &quot;HA_PROJECT&quot;.</td>
</tr>
<tr>
<td>STREAMING_CONNECTION_ID</td>
<td>BIGINT</td>
<td></td>
<td>A unique integer ID for the connection.</td>
</tr>
<tr>
<td>STREAM_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name of the stream subscribed to.</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The user name provided by the client during connection establishment, displayed once the user is authenticated.</td>
</tr>
<tr>
<td>CLIENT_HOST</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The symbolic host name of the client machine, if available. If not available, host is the IP address of the client machine.</td>
</tr>
<tr>
<td>CLIENT_IP</td>
<td>VARCHAR(64)</td>
<td></td>
<td>The IP address of the client machine, as a string.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------</td>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CLIENT_PORT</td>
<td>INTEGER</td>
<td></td>
<td>The TCP port number from which the connection originates.</td>
</tr>
<tr>
<td>CONNECTION_TAG</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The user-set symbolic connection tag name. If not set by the user, the connection tag is NULL.</td>
</tr>
<tr>
<td>IS_SUBSCRIBED</td>
<td>VARCHAR(5)</td>
<td></td>
<td>The status of a subscription to a stream. TRUE if subscribed, FALSE if not subscribed.</td>
</tr>
<tr>
<td>TRANSACTION_THROUGHPUT</td>
<td>REAL</td>
<td>MONITOR_UNIT_RATE</td>
<td>The client’s performance, in transactions per second, received by the client since the last update.</td>
</tr>
<tr>
<td>ROW_THROUGHPUT</td>
<td>REAL</td>
<td>MONITOR_UNIT_RATE</td>
<td>The client’s performance, in data rows per second, received by the client since the last update.</td>
</tr>
<tr>
<td>INCREMENTAL_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The number of transactions, envelopes, and messages received by the client since the last update.</td>
</tr>
<tr>
<td>INCREMENTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The number of data rows received by the client since the last update.</td>
</tr>
<tr>
<td>TOTAL_TRANSACTION_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The total number of transactions, envelopes, and messages received by the client.</td>
</tr>
<tr>
<td>TOTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The total number of data rows received by the client.</td>
</tr>
<tr>
<td>TOTAL_ROWS_DROPPED_COUNT</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_COUNTER</td>
<td>The total number of data rows dropped in the gateway because they were not read quickly enough by the client. For lossy subscriptions.</td>
</tr>
<tr>
<td>ACCUMULATOR_SIZE</td>
<td>INTEGER</td>
<td></td>
<td>The number of rows collected in the accumulator to be sent in the next pulse. For pulsed subscriptions.</td>
</tr>
<tr>
<td>QUEUE_SIZE</td>
<td>INTEGER</td>
<td></td>
<td>The number of rows queued for transmission to the client.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------</td>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QUEUE_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>The subscriber queue size, as a percentage, relative to the queue size limit. If queue utilization reaches 100 percent, any subsequent attempts to post data to this client are blocked, propagating the flow control back to the source of the post.</td>
</tr>
<tr>
<td>WORK_QUEUE_SIZE</td>
<td>INTEGER</td>
<td></td>
<td>The number of rows queued for transmission to the client. These rows have been transferred from the internal queue to the socket buffer. The rows can be regrouped by envelopes.</td>
</tr>
<tr>
<td>CPU_CORE_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>Total CPU usage for the client thread, as a percentage of a single CPU core.</td>
</tr>
<tr>
<td>CPU_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>Total CPU usage for the client, as a percentage of all CPU cores on the machine. Total CPU usage equals system CPU usage plus user CPU usage.</td>
</tr>
<tr>
<td>CPU_SYSTEM_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>System CPU usage for the client, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>CPU_USER_UTILIZATION</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>User CPU usage for the client, as a percentage of all CPU cores on the machine.</td>
</tr>
<tr>
<td>TOTAL_CPU_TIME</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Total CPU time since the creation of the client, in microseconds. Total CPU time equals system CPU time plus user CPU time.</td>
</tr>
<tr>
<td>TOTAL_CPU_SYSTEM_TIME</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Total system CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>TOTAL_CPU_USER_TIME</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Total user CPU time, in microseconds, since the creation of the client thread.</td>
</tr>
<tr>
<td>CONNECT_TIME</td>
<td>TIMESTAMP</td>
<td></td>
<td>The time when the server accepts (but does not authenticate) the connection.</td>
</tr>
<tr>
<td>SUBSCRIBER_DURATION</td>
<td>BIGINT</td>
<td>MONITOR_UNIT_US</td>
<td>Duration of lapsed real time since the creation of the client thread.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Unit</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LAST_UPDATE_TIME</td>
<td>TIMESTAMP</td>
<td></td>
<td>The most recent time that statistics were updated for the subscriber.</td>
</tr>
</tbody>
</table>

### 7.2.8 M_STREAMING_PROJECT_STREAMS System View

**[Streaming Analytics]**

Provides information and statistics about each stream in an SAP HANA streaming analytics project.

Only users with a MONITOR role can view the content of this system view. The view is empty for all other users.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The schema in which the project is deployed.</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name given to the project. It corresponds to the APPLICATION_NAME in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE &quot;PROJECT&quot; or &quot;HA_PROJECT&quot;.</td>
</tr>
<tr>
<td>PROJECT_INSTANCE</td>
<td>INTEGER</td>
<td></td>
<td>The instance number of the project. It corresponds to the APPLICATION_INSTANCE in the M_STREAMING_APPLICATIONS System View for APPLICATION_TYPE &quot;PROJECT&quot; or &quot;HA_PROJECT&quot;.</td>
</tr>
<tr>
<td>STREAM_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name of the stream</td>
</tr>
<tr>
<td>STREAM_TYPE</td>
<td>VARCHAR(32)</td>
<td></td>
<td>The type of the stream.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• STREAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• DELTASTREAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• WINDOW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• METADATA</td>
</tr>
</tbody>
</table>

SAP HANA SQL Reference Guide for SAP HANA Platform

System Views Reference for Additional SAP HANA Contexts
<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VISIBILITY</td>
<td>VARCHAR(32)</td>
<td></td>
<td>The visibility of the stream. Visibilities include: input, output, local, and intermediate.</td>
</tr>
<tr>
<td>TARGET_NAME</td>
<td>VARCHAR(256)</td>
<td></td>
<td>Denotes that this stream is an intermediate stream used to populate this target.</td>
</tr>
<tr>
<td>GD_SUPPORT_LEVEL</td>
<td>VARCHAR(32)</td>
<td></td>
<td>Indicates whether guaranteed delivery (GD) is enabled for the stream or window.</td>
</tr>
<tr>
<td>TRANSACTION_THROUGHPUT</td>
<td>REAL</td>
<td></td>
<td>The stream's performance, in transactions per second, since the last update.</td>
</tr>
<tr>
<td>ROW_THROUGHPUT</td>
<td>REAL</td>
<td></td>
<td>The stream's performance, in rows per second, since the last update.</td>
</tr>
<tr>
<td>INCREMENTAL_TRANS_COUNT</td>
<td>BIGINT</td>
<td></td>
<td>The number of transactions processed by the server since the last update.</td>
</tr>
<tr>
<td>INCREMENTAL_ROW_COUNT</td>
<td>BIGINT</td>
<td></td>
<td>The number of rows processed by the server since the last update.</td>
</tr>
<tr>
<td>QUEUE_SIZE</td>
<td>INTEGER</td>
<td></td>
<td>The stream's input queue size.</td>
</tr>
<tr>
<td>STORE_ROW_COUNT</td>
<td>BIGINT</td>
<td></td>
<td>The current number of records in the stream's store.</td>
</tr>
<tr>
<td>CPU_CORE_UTILIZATION</td>
<td>REAL</td>
<td></td>
<td>Total CPU usage for the stream thread, as a percentage of a single CPU core.</td>
</tr>
<tr>
<td>MONITOR_UNIT_RATE</td>
<td>REAL</td>
<td>MONITOR_UNIT_PERCENT</td>
<td>The stream's performance, in transactions per second, since the last update.</td>
</tr>
<tr>
<td>MONITOR_UNIT_COUNTER</td>
<td>BIGINT</td>
<td></td>
<td>The number of rows processed by the server since the last update.</td>
</tr>
<tr>
<td>MONITOR_UNIT_UPDATE</td>
<td>BIGINT</td>
<td></td>
<td>The sequence number of the current update.</td>
</tr>
<tr>
<td>SEQUENCE_NUMBER</td>
<td>BIGINT</td>
<td></td>
<td>The sequence number of the current update.</td>
</tr>
<tr>
<td>STORE_ROW_COUNT</td>
<td>BIGINT</td>
<td></td>
<td>The current number of records in the stream's store.</td>
</tr>
<tr>
<td>QUEUE_SIZE</td>
<td>INTEGER</td>
<td></td>
<td>The stream's input queue size.</td>
</tr>
<tr>
<td>UNIT</td>
<td></td>
<td></td>
<td>The visibility of the stream. Visibilities include: input, output, local, and intermediate.</td>
</tr>
</tbody>
</table>
### M_STREAMING_SCHEMAS System View [Streaming Analytics]

Provides a list of the streaming schemas in the SAP HANA streaming analytics cluster

Only users with a MONITOR role can view the content of this system view. The view is empty for all other users.

#### Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA_NAME</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>The name of the schema where the streaming applications are deployed.</td>
</tr>
</tbody>
</table>
7.2.10  M_STREAMING_SERVICES System View [Streaming Analytics]

Provides a list of logical node names in the SAP HANA streaming analytics cluster.

Only users with a MONITOR role can view the content of this system view. The view is empty for all other users.

Structure

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVICE_ALIAS</td>
<td>NVARCHAR(256)</td>
<td></td>
<td>Name of the streaming node.</td>
</tr>
<tr>
<td>HOST</td>
<td>VARCHAR(64)</td>
<td></td>
<td>Streaming node host.</td>
</tr>
<tr>
<td>PORT</td>
<td>INTEGER</td>
<td></td>
<td>Streaming node port.</td>
</tr>
</tbody>
</table>

7.2.11  Additional Views For Streaming Analytics

Views including information pertaining to, but not limited to, SAP HANA streaming analytics.

<table>
<thead>
<tr>
<th>View</th>
<th>Information Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANTED_PRIVILEGES [page 1580]</td>
<td>Privileges granted to users and roles</td>
</tr>
<tr>
<td>GRANTED_ROLES [page 1581]</td>
<td>Roles granted to users or other roles</td>
</tr>
<tr>
<td>M_HOST_RESOURCE_UTILIZATION [page 1937]</td>
<td>Host resource utilization, CPU time is in milliseconds and added across all cores since system start</td>
</tr>
<tr>
<td>M_SERVICE_MEMORY [page 2113]</td>
<td>Detailed information on memory utilization by services</td>
</tr>
<tr>
<td>M_SERVICE_STATISTICS [page 2122]</td>
<td>Statistics on active services</td>
</tr>
<tr>
<td>M_SERVICE_TYPES [page 2132]</td>
<td>Service types</td>
</tr>
<tr>
<td>M_SERVICES [page 2107]</td>
<td>Status of all services</td>
</tr>
<tr>
<td>ROLES [page 1646]</td>
<td>Shows available roles</td>
</tr>
<tr>
<td>USERS [page 1698]</td>
<td>All users</td>
</tr>
</tbody>
</table>
8  Important Disclaimer for Features in SAP HANA

For information about the capabilities available for your license and installation scenario, refer to the Feature Scope Description for SAP HANA.
Index

Special Characters
[NOT] MEMBER OF 73

A
ABS 91
ABSTRACT_SQLPlans 1477
Access Control Statements 1199
ACCESSIBLE_VIEWS 1480
ACOS 92
ADAPTER_CAPABILITIES 1481
ADAPTER_LOCATIONS 1482
ADAPTERS 1480
ADD_DAYS 93
ADD_MONTHS 94
ADD_MONTHS_LAST 95
ADD_SECONDS 97
ADD_WORKDAYS 98
ADD_YEARS 102
AFL_AREAS 1483
AFL_FUNCTION_PARAMETERS 1485
AFL_FUNCTION_PROPERTIES 1486
AFL_FUNCTIONS 1484
AFL_PACKAGES 1487
AFL_TEXTS 1488
AGENT_CONFIGURATION 1489
AGENT_GROUPS 1490
AGENTS 1488
Aggregate Functions 414
ALTER_AUDIT_POLICY 431
ALTER_CREDENTIAL 439
ALTER_DATABASE 440
ALTER_FUNCTION 447
ALTER_INDEX 452
ALTER_PROCEDURE 463
ALTER_PROCEDURE_RECOMPILE 463
ALTER_PSE 468
ALTER_REMOTE_SOURCE 471
ALTER_ROLE 475
ALTER_SAML_PROVIDER 476
ALTER_SEQUENCE 479
ALTER_STATISTICS 484
ALTER_SYSTEM_ADD_ABSTRACT_SQL_PLAN_FILTER 497
ALTER_SYSTEM_ADD_STATEMENT_HINT 493
ALTER_SYSTEMALTER_CONFIGURATION 500
ALTER_SYSTEMALTER_DATAVOLUME_ADD_PARTITION 503
ALTER_SYSTEMALTER_DATAVOLUME_DROP_PARTITION 504
ALTER_SYSTEMALTER_SESSION_SET 506
ALTER_SYSTEMALTER_SESSION_UNSET 507
ALTER_SYSTEMALTER_STATEMENT_HINT 493
ALTER_SYSTEMALTER_TABLEPlacement 508
ALTER_SYSTEMAPPLICATION_ENCRYPTION 512
ALTER_SYSTEMCANCEL_WORK_IN_SESSION 516
ALTER_SYSTEMCLEAR_AUDIT_LOG 517
ALTER_SYSTEMCLEAR_CACHE 518
ALTER_SYSTEMCLEAR_COLUMNJOIN_DATA_STATISTICS 520
ALTER_SYSTEMCLEAR_INIFILECONTENT_HISTORY 521
ALTER_SYSTEMCLEAR_RESULT_CACHE 522
ALTER_SYSTEMCLEAR_SQL_PLAN_CACHE 522
ALTER_SYSTEMCLEAR_SQLSCRIPT_PLAN_PROFILER 576
ALTER_SYSTEMCLEAR_TIMEZONE_CACHE 523
ALTER_SYSTEMCLEAR_TRACES 525
ALTER_SYSTEMDISABLE_ABSTRACT_SQL_PLAN 532
ALTER_SYSTEMDISABLE_ALLASYNC_TABLES 534
ALTER_SYSTEMDISABLE_ALL_SYNCHRONOUS_TABLES 534
ALTER_SYSTEMDISABLE_STATEMENT_HINT 535
ALTER_SYSTEMDISABLE_SYSTEM_REPLICATION 537
ALTER_SYSTEMDISCONNECT_SESSION 531
ALTER_SYSTEMENABLE_ABSTRACT_SQL_PLAN 532
ALTER_SYSTEMENABLE_ALLASYNC_TABLES 534
ALTER_SYSTEMENABLE_ALL_SYNCHRONOUS_TABLES 534
ALTER_SYSTEMENABLE_STATEMENT_HINT 535
ALTER_SYSTEMENABLE_SYSTEM_REPLICATION 537
ALTER_SYSTEMLOAD_PERFTRACE 539
ALTER_SYSTEMLOGGING 543
ALTER_SYSTEMMIGRATE_ABSTRACT_SQL_PLAN 544
ALTER_SYSTEMPERSISTENCE_ENCRYPTION 545
ALTER_SYSTEMPIN_SQL_PLAN_CACHEENTRY 548
ALTER_SYSTEMRECLAIMDATASPACE 550
ALTER_SYSTEMRECLAIMDATAVOLUME 551
ALTER_SYSTEMRECLAIMFLEXIBLETABLES 552
ALTER_SYSTEMRECLAIMLOG 553
ALTER_SYSTEMRECLAIMVERSIONSPACE 554

SAP HANA SQL Reference Guide for SAP HANA Platform
ALTER SYSTEM RECOMPILE SQL PLAN CACHE ENTRY 555
ALTER SYSTEM RECONFIGURE SERVICE 556
ALTER SYSTEM REFRESH RESULT CACHE 557
ALTER SYSTEM REFRESH RESULT CACHE ENTRY 558
ALTER SYSTEM REMOVE ABSTRACT SQL PLAN 532
ALTER SYSTEM REMOVE ABSTRACT SQL PLAN FILTER 497
ALTER SYSTEM REMOVE RESULT CACHE ENTRY 561
ALTER SYSTEM REMOVE SQL PLAN CACHE ENTRY 562
ALTER SYSTEM REMOVE STATEMENT HINT 493
ALTER SYSTEM REMOVE TRACES 564
ALTER SYSTEM RESET MONITORING VIEW 565
ALTER SYSTEM SAVE PERFTRACE 566
ALTER SYSTEM SAVEPOINT 567
ALTER SYSTEM START CAPTURE ABSTRACT SQL PLAN 575
ALTER SYSTEM START DATABASE 583
ALTER SYSTEM START PERFTRACE 578
ALTER SYSTEM START SQLSCRIPT CODE COVERAGE 580
ALTER SYSTEM START SQLSCRIPT PLAN PROFILER 576
ALTER SYSTEM STOP APPLY ABSTRACT SQL PLAN 570
ALTER SYSTEM STOP DATABASE 584
ALTER SYSTEM STOP SERVICE 585
ALTER SYSTEM STOP SQLSCRIPT PLAN PROFILER 576
ALTER SYSTEM UNPIN SQL PLAN CACHE ENTRY 548
ALTER SYSTEM UPDATE ABSTRACT SQL PLAN 586
ALTER TABLE 589
ALTER USER 654
ALTER VIEW 667
ALTER VIRTUAL TABLE 676
ALTER WORKLOAD CLASS 678
ALTER WORKLOAD MAPPING 681
ANNOTATE 686
ANNOTATIONS 1494
APPLICATION_CONNECTION_HISTORY 1497
APPLICATION_ENCRYPTION_KEYS 1498
Array Functions 415
ASCII 104
ASIN 105
ASSOCIATIONS 1499
ATAN 106
ATAN2 107

AUDIT_ACTIONS 1500
AUDIT_LOG 1501
AUDIT_POLICIES 1503
AUTHORIZATION_GRAPH 1506
AUTHORIZATION_OBJECTS 1508
AUTHORIZATION_TYPES 1509
AUTO_CORR 108
AVG 110

BACKUP CANCEL 688
BACKUP CATALOG DELETE 690
BACKUP CHECK 695
BACKUP DATA 699
BACKUP DATA CLOSE SNAPSHOT 704
BACKUP DATA CREATE SNAPSHOT 706
BACKUP DATA DROP SNAPSHOT 708
Backup Statements 1200
BINNING Function 112
BINTOHEX 114
BINTONHEX 115
BINTOSTR 116
BITAND 117
BITCOUNT 118
BITNOT 119
BITOR 120
BITSET 121
BITUNSET 122
BITXOR 123

CALL 715
CARDINALITY 124
CAST 125
CDS_ANNOTATION_ASSIGNMENTS 1510
CDS_ANNOTATION_VALUES 1511
CDS_ARTIFACT_NAMES 1512
CDS_ASSOCIATIONS 1513
CDS_ENTITIES 1514
CDS_VIEWS 1515
CEIL 126
CERTIFICATES 1516
CHAR 127
CLIENTSIDE_ENCRYPTION_COLUMN_KEYS 1517
COALESCE 128
COLLABORATIONS 1520
COLLATIONS 1520
COMMENT ON 718
COMMIT 720
Comparison Predicates 62
CONCAT 129
CONCAT_NZ 130
CONNECT 721
CONSTRAINTS 1523
CONTAINS Predicate 64
CONVERT_CURRENCY 131
CONVERT_INTERNAL_TYPE_INTO_NAME 1524
CONVERT_UNIT 142
CORR 145
CORR_SPEARMAN 148
COS 153
COSH 153
COT 154
COUNT 155
CREATE AUDIT POLICY 722
CREATE CERTIFICATE 739
CREATE CREDENTIAL 744
CREATE FULLTEXT INDEX 748
CREATE FUNCTION 752
CREATE GRAPH WORKSPACE 760
CREATE INDEX 762
CREATE PROCEDURE 776
CREATE PSE 797
CREATE REMOTE SOURCE 799
CREATE ROLE 802
CREATE SAML PROVIDER 803
CREATE SCHEMA 807
CREATE SEQUENCE 810
CREATE SEQUENCES 811
CREATE STATISTICS 815
CREATE SYNONYM 823
CREATE TRIGGER 876
CREATE TYPE 892
CREATE VIEW 904
CREATE VIEW 909
CREATE CTE 917
CREATE VIRTUAL FUNCTION 917
CREATE WORKLOAD CLASS 929
CREATE WORKLOAD MAPPING 934
CREDENTIALS 1524
CS_ALL_COLUMNS 1525
CS_BO_VIEWS 1526
CS_CONCAT_COLUMNS 1527
CS_FREESTYLE_COLUMNS 1528
CS_JOIN_CONDITIONS 1529
CS_JOIN_CONSTRAINTS 1530
CS_JOIN_PATHS 1531
CS_JOIN_TABLES 1532
CS_KEY_FIGURES 1533
CS_VIEW_COLUMNS 1535
CUBIC_SPLINE_APPROX Function 159
CUME_DIST Function 164
CURRENT_CONNECTION 166
CURRENT_DATE 167
CURRENT_IDENTITY_VALUE 168
CURRENT_MVCC_SNAPSHOT_TIMESTAMP 168
CURRENT_OBJECT_SCHEMA 169
CURRENT_SCHEMA 170
CURRENT_TIME 171
CURRENT_TIMESTAMP 172
CURRENT_TRANSACTION_ISOLATION_LEVEL 173
CURRENT_UPDATE_STATEMENT_SEQUENCE 174
CURRENT_UPDATE_TRANSACTION 175
CURRENT_USER 175
CURRENT_UTCDATE 177
CURRENT_UTCTIME 178
CURRENT_UTCTIMESTAMP 179

D

Data Definition Statements 1201
Data Import Export Statements 1202
Data Manipulation Statements 1203
Data Type Conversion Functions 416
DATA_STATISTICS 1538, 1851
DATA_TYPES 1542
Date/Time Functions 417
DAYNAME 180
DAYOFMONTH 180
DAYOFYEAR 181
DAYS_BETWEEN 182
DELETE 938
DEPENDENCY_RULE_COLUMNS 1545
DEPENDENCY_RULES 1544
DFT 185
DO BEGIN...END 942
DROP AUDIT POLICY 946
DROP CERTIFICATE 948
DROP CREDENTIAL 953
DROP DATABASE 954
DROP FULLTEXT INDEX 955
DROP FUNCTION 956
DROP GRAPH WORKSPACE 957
DROP INDEX 958
DROP PROCEDURE 963
DROP PSE 964
DROP REMOTE SOURCE 966
DROP ROLE 967
DROP SAML PROVIDER 969
DROP SCHEMA 971
| JSON_TABLE 223 |
| JSON_VALUE 231 |
| LAG Function 234 |
| LANGUAGE 236 |
| LAST_DAY 237 |
| LAST_VALUE 238 |
| LCASE 239 |
| LCM_PRODUCT_INSTANCES 1593 |
| LCM_PRODUCT_INSTANCES_INCLUDED 1594 |
| LCM_PRODUCTS 1592 |
| LCM_SOFTWARE_COMPONENTS 1595 |
| LCM_SWID 1596 |
| LEAD Function 240 |
| LEAST 242 |
| LEFT 242 |
| LENGTH 243 |
| LIBRARIES 1600 |
| LIKE Predicate 71 |
| LIKE_REGEXPR 72 |
| LINEAR_APPROX Function 244 |
| LN 248 |
| LOAD 1054 |
| LOCALTOUTC 249 |
| LOCATE 250 |
| LOCATE_REGEXPR 252 |
| LOCK TABLE 1056 |
| LOG 254 |
| LOWER 255 |
| LPAD 255 |
| LTRIM 257 |
| M_ABSTRACT_SQL_PLAN_OVERVIEW 1730 |
| M_ABSTRACT_SQL_PLAN_STATISTICS 1731 |
| M_ACTIVE_PROCEDURES 1733 |
| M_ACTIVE_STATEMENTS 1735 |
| M_ADMISSION_CONTROL_EVENTS 1739 |
| M_ADMISSION_CONTROL_QUEUES 1740 |
| M_ADMISSION_CONTROL_STATISTICS 1741 |
| M_AFL_FUNCTIONS 1742 |
| M_AFL_STATES 1743 |
| M_AGENTS 1744 |
| MASYNCHRONOUS_TABLE_REPLICAS 1746 |
| MASYNCHRONOUS_TABLE_REPLICAS_RESET 1747 |
| M_ATTACHED_STORAGES 1747 |
| M_BACKUP_CATALOG 1748 |
| M_BACKUP_CATALOG_FILES 1751 |
| M_BACKUP_CONFIGURATION 1754 |
| M_BACKUP_PROGRESS 1758 |
| M_BACKUP_SIZE_ESTIMATIONS 1759 |
| M_BLOCKED_TRANSACTIONS 1761 |
| M_BUFFER_CACHE_POOL_STATISTICS 1762 |
| M_BUFFER_CACHE_POOL_STATISTICS_RESET 1764 |
| M_BUFFER_CACHE_STATISTICS 1764 |
| M_BUFFER_CACHE_STATISTICS_RESET 1765 |
| M_CACHE_ENTRIES 1768 |
| M_CACHES 1766 |
| M_CACHES_RESET 1767 |
| M_CATALOG_MEMORY 1769 |
| M_CE_CALCSCENARIOS 1770 |
| M_CE_CALCSCENARIOS_OVERVIEW 1771 |
| M_CE_CALCVIEW_DEPENDENCIES 1772 |
| M_CE_DEBUG_INFOS 1773 |
| M_CE_DEBUG_JSONS 1774 |
| M_CE_DEBUG_NODE_MAPPING 1775 |
| M_CE_PLE_CALCSCENARIOS 1776 |
| M_CLIENT_VERSIONS 1777 |
| M_COLLECTION_TABLE_VIRTUAL_FILES 1779 |
| M_COLLECTIONS_TABLES 1778 |
| M_COMPACTION_THREAD 1780 |
| M_CONDITIONAL_VARIABLES 1782 |
| M_CONDITIONAL_VARIABLES_RESET 1784 |
| M_CONFIGURATION_PARAMETERS 1521 |
| M_CONNECTION_STATISTICS 1790 |
| M_CONNECTIONS 1785 |
| M_CONTAINER_DIRECTORY 1799 |
| M_CONTAINER_NAME_DIRECTORY 1800 |
| M_CONTEXT_MEMORY 1801 |
| M_CONTEXT_MEMORY_RESET 1804 |
| M_CONVERTER_STATISTICS 1805 |
| M_CONVERTER_STATISTICS_RESET 1807 |
| M_CS_ALL_COLUMNS 1807 |
| M_CS_COLUMNS 1814 |
| M_CS_COLUMNS_PERSISTENCE 1818 |
| M_CS_INDEXES 1819 |
| M_CS_LOADS 1821 |
| M_CS_LOB_SPACE_RECLAIMS 1822 |
| M_CS_MVCC 1826 |
| M_CS_PARTITION 1833 |
| M_CS_RECORD_LOCK_STATISTICS 1834 |
| M_CS_TABLES 1837 |
| M_CS_UNLOADS 1842 |
| M_CUSTOMIZABLE_FUNCTIONALITIES 1844 |
| M_DATA_VOLUME_PAGE_STATISTICS 1855 |
| M_DATA_VOLUME_PAGE_STATISTICS_RESET 1857 |
| M_DATA_VOLUME_SUPERBLOCK_STATISTICS 1862 |
| M_DATA_VOLUMES 1854 |
Notation 30
NOW 278
NTH_VALUE 279
NTILE Function 281
NULL Predicate 74
NULLIF 282
Number Functions 421
NUMTOHEX 283

O
OBJECT_DEPENDENCIES 1604
OBJECT_PRIVILEGES 1605
OBJECTS 1603
OCCURRENCES_REGEXPR 284
Operators 51
OWNERSHIP 1606

P
PERCENT_RANK Function 286
PINNED_SQL_PLANS 1609
PLAINTEXT 291
POWER 292
Predicates 61
PRIVILEGES 1611
Procedural Statements 1204
PROCEDURE_OBJECTS 1614
PROCEDURE_PARAMETERS 1615
PROCEDURE_ROUTES 1617
PROCEDURES 1612
PROJECTION_VIEW_COLUMN_SOURCES 1618
PSE_CERTIFICATES 1620
PSES 1619

Q
QUARTER 292
QUERY_PLANS 1624

R
RAND 293
RAND_SECURE 297
RANDOM_PARTITION Function 294
Range Predicate 63
RANK Function 298
RECOVER DATA 1064
REFERENTIAL_CONSTRAINTS 1625
REFRESH STATISTICS 1082

REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINTS 1630
REMOTE_SOURCE_OBJECT_COLUMN_DEFINITIONS 1631
REMOTE_SOURCE_OBJECT_COLUMNS 1629
REMOTE_SOURCE_OBJECT_DESCRIPTIONS 1632
REMOTE_SOURCE_OBJECTS 1627
REMOTE_SOURCES 1626
REMOTE_SUBSCRIPTION_DATA_CONTAINERS 1635
REMOTE_SUBSCRIPTION.Exceptions 1636
REMOTE_USERS 1637
RENAME COLUMN 1088
RENAME DATABASE 1090
RENAME INDEX 1091
RENAME TABLE 1094
REORG_GENERATE_OVERVIEW 1638
REORG_OVERVIEW 1639
REORG_PLAN 1640
REORG_PLAN_INFOS 1641
REORG_STEPS 1642
REPLACE 302, 1096
REPLACE_REGEXPR 303
RESERVED_KEYWORDS 1643
Restrictions for the database 1208
RESULT_CACHE 1644
RESULT_CACHE_COLUMNS 1645
RESULT_CACHE_ID 305
RESULT_CACHE_REFRESH_TIME 306
REVOKE 1098
RIGHT 307
ROLES 1646
ROLLBACK 1101
ROUND 308
ROW_NUMBER Function 312
RPAD 310
RTRIM 311

S
SAML_PROVIDERS 1648
SAML_USER_MAPPINGS 1649
SAP Data Types 33
SCHEMAS 1651
SCORE 199, 314
SEARCH_RULE_SET_CONDITIONS 1653
SEARCH_RULE_SETS 1652
SECOND 319
SECONDS_BETWEEN 320
SELECT 1104
SEQUENCES 1654
Series Data Functions 422
SERIES_DISAGGREGATE 320
SERIES_ELEMENT_TO_PERIOD 325
SERIES_GENERATE 331
SERIES_KEY_COLUMNS 1655
SERIES_PERIOD_TO_ELEMENT 335
SERIES_ROUND 337
SERIES_TABLES 1656
Session Management Statements 1205
SESSION_CONTEXT 341
SESSION_COOKIES 1658
SESSION_USER 342
SET [SESSION] 1171
SET HISTORY SESSION 1165
SET PSE 1167
SET SCHEMA 1170
SET SYSTEM LICENSE 1172
SET TRANSACTION 1173
SET TRANSACTION AUTOCOMMIT DDL 1176
SIGN 343
SIN 344
SINH 344
SOUNDEX 345
SQL Error Codes 1212
SQL Functions 83
SQL Reference 27
SQL Statements 430
SQLSCRIPT_ANALYZER_RULES 1659
SORT 346
ST_GEOMETRY_COLUMNS 1664
ST_SPATIAL_REFERENCE_SYSTEMS 1665
ST_UNITS_OF_MEASURE 1668
STATEMENT_HINTS 1661
STDDE 347
STDDEV_POP 349
STDDEV_SAMP 350
String Functions 423
STRING_AGG 351
STRTOBIN 353
STRUCTURED_PRIVILEGES 1662
SUBARRAY 353
SUBSTR_AFTER 355
SUBSTR_BEFORE 356
SUBSTRING 359
SUBSTRING_REGEXP 357
SUM 360
SYNONYMS 1668
System Management Statements 1205
System Views Reference 1476
SYUUUI 362
T
TABLE_COLUMNS 1675
TABLE_COLUMNS_ODBC 1679
TABLE_GROUPS 1680
TABLE_PLACEMENT 1684
TABLE_PLACEMENT_LOCATIONS 1685
TABLE_REPLICAS 1686
TABLES 1669
TAN 362
TANH 363
TASK_PARAMETERS 1689
TASKS 1687
TEMPORAL_TABLES 1690
Tenant Database Management Statements 1207
TO_ALPHANUM 364
TO_BIGINT 364
TO_BINARY 365
TO_BLOB 366
TO_CLOB 368
TO_DATE 368
TO_DATS 369
TO_DECIMAL 370
TO_DOUBLE 371
TO_FIXEDCHAR 372
TO_INT 373
TO_INTEGER 373
TO_NCHAR 375
TO_NVARCHAR 376
TO_REAL 377
TO_SECONDDATE 378
TO_SMALLDECIMAL 379
TO_SMALLINT 380
TO_TIME 380
TO_TIMESTAMP 381
TO_TINYINT 382
TO_VARCHAR 383
Transaction Management Statements 1208
TRANSACTION_HISTORY 1693
TRIGGERS 1694
TRIM 385
TRIM_ARRAY 386
TRUNCATE TABLE 1178
U
UCASE 387
UMINUS 387
UNICODE 388
UNLOAD 1179
UNSET [SESSION] 1183
UNSET PSE 1181
UNSET SYSTEM LICENSE ALL 1184
UPDATE 1185
UPPER 389
UPSERT 1191
USER_PARAMETERS 1702
USERGROUP_PARAMETERS 1697
USERGROUPS 1696
USERS 1698
UTCTOLOCAL 390

V
VAR 391
VAR_POP 393
VAR_SAMP 394
VIEW_COLUMNS 1705
VIEW_PARAMETERS 1709
VIEWS 1702
VIRTUAL_COLUMNS 1710
VIRTUAL_FUNCTION_PACKAGES 1713
VIRTUAL_FUNCTIONS 1712
VIRTUAL_TABLES 1718

W
WEEK 395
WEEKDAY 396
WEIGHTED_AVG Function 396
WIDTH_BUCKET 398
window aggregate functions 428
Window Functions 424
WORKDAYS_BETWEEN 400
Workload Management Statements 1208
WORKLOAD_CLASSES 1721
WORKLOAD_MAPPINGS 1723

X
X509_USER_MAPPINGS 1726
XML EXTRACT 404
XML EXTRACT VALUE 405
XML TABLE 407
XSA_AUDIT_LOG 1727

Y
YEAR 413
YEARS_BETWEEN 413
Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information. About the icons:

- Links with the icon ![link icon]: You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP’s gross negligence or willful misconduct.

- Links with the icon ![link icon]: You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.

The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP’s gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.