ADMINISTRATION GUIDE | INTERNAL
SAP IQ 16.1 SP 03
Document Version: 1.0.0 – 2018-11-20

SAP IQ Administration: Database

© 2020 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

Content

1 SAP IQ Administration: Database. 14

2 Database Administration Workflow. 15

3 Run SAP IQ Database Servers. 16

3.1 Starting Servers. 16

The Server Startup Utility. 18

Starting the Server Using the Startup Utility. 19

Starting a Server from Interactive SQL. 19

Automatic Server Startup. 20

3.2 Command Line Options. 20

Displaying Command Line Options. 21

Required Command Line Options. 21

Server and Database Names. 21

Performance Options. 23

Privilege Options. 25

Client/Server Environment Options. 25

Catalog Page Size Option. 26

3.3 Multiple Servers on a System. 27

3.4 Configuration Files. 27

Configuration File List. 28

Examples of Configuration Files. 28

Default Configuration File. 29

Configuration File for the Demo Database. 29

3.5 Server Activity Logs. 29

Naming the Server Log File. 32

UNIX Log Files. 32

3.6 Stopping Servers. 32

Operating System Session Shutdown. 35

When to Stop and Restart the Server. 35

3.7 Starting and Stopping Databases. 36

Database Startup Guidelines. 36

Stopping Databases. 37

3.8 Starting the iqdemo Database. 37

4 Connect to SAP IQ Servers and Databases. 38

4.1 Ways to Connect. 39

Connecting to the Demo Database from dbisql (UNIX). 41

Connecting to the Demo Database from dbisql (Windows). 42
How Database Status Affects Local Connections. 42
Connecting to an Embedded Database. 43
TCP/IP protocol. 44
Connecting Using a Data Source. 48
Connecting from SAP IQ Utilities. 49
Default Connection Parameters. 49
4.2 Supported Connection Interfaces. 50
4.3 Connection Status. 51
4.4 How Connection Parameters Work. 52
Format for Connection Strings. 53
How Applications Pass Connection Parameters. 53
4.5 Connection Parameters in ODBC Data Sources. 54
4.6 Drivers for Connections. 55
4.7 File Data Sources. 56
Creating a File Data Source Using the ODBC Administrator. 57
File Data Sources and Text Editors. 57
4.8 Connections from Other Databases. 58
Avoiding Port Number Conflicts on UNIX. 59
4.9 How to Test Connections. 59
4.10 Integrated Logins. 60
Using Integrated Logins. 62
Security Concerns: Unrestricted Database Access. 64
Temporary Public Options Provide Added Security. 65
Network Aspects of Integrated Logins. 66
Default Integrated Login Users. 66
4.11 Connection Pooling. 66
4.12 Temporary Connections. 68
4.13 Logical Server Configuration. 69
Connections in Simplex. 70
Connections in Multiplex. 70
4.14 How to End Connections. 71
4.15 Connection Logging. 72

5 Create and Manage SAP IQ Databases. 73
5.1 SAP IQ Database Design. 73
5.2 Create SAP IQ Databases. 74
Database Creation with SQL. 75
IQ Main Store and IQ Temporary Store Space Management. 85
5.3 Dropping a Database. 86
5.4 Schema and Data Export. 86
iqunload Schema and Data Export Utility. 87

Load Schema and Data. . . . 91
5.5 Disconnecting Other Users from a Database. . . . 92
5.6 Disconnecting from a Database in Embedded SQL. . . . 92
5.7 Disconnecting All Connections from a Database in Interactive SQL. . . . 93
5.8 Showing System Objects in Interactive SQL. . . . 93
5.9 Setting Database Options in Interactive SQL. . . . 94

6 Manage Data Storage in SAP IQ. . . . 95
6.1 Space Allocation. . . . 96
6.2 Types of Dbspaces. . . . 96
Catalog Store. . . . 98
IQ_SYSTEM_MAIN Dbspace. . . . 99
Other User Main Dbspaces. . . . 99
IQ Temporary Dbspace. . . . 99
IQ Shared Temporary Dbspace. . . . 100
IQ Message File Dbspace. . . . 101
6.3 Space for Databases. . . . 101
6.4 Space Requirements for IQ Stores and Temporary Stores. . . . 102
6.5 Sizing Guidelines for Main and Temporary Stores. . . . 102
6.6 Setting Up Windows Access to Raw Devices. . . . 104
6.7 Choosing Drives for Raw Devices on Windows. . . . 105
6.8 Viewing Access Permissions of a Raw Device. . . . 106
6.9 Setting Permissions for a Raw Device. . . . 107
6.10 Estimating Space and Dbspaces Required. . . . 108
6.11 Dbspace Management Example. . . . 109
6.12 Data Storage. . . . 114
Dbfile Attributes and Operations. . . . 115
Dbspace Attributes and Operations. . . . 118
Read-Only and Read-Write Dbspaces and Files. . . . 121
6.13 Cache Dbspace for Direct-Attached Storage. . . . 126
Creating the Cache Dbspace. . . . 128
Dropping the Cache Dbspace. . . . 129
Adding Storage to the Cache Dbspace. . . . 129
Removing Storage from the Cache Dbspace. . . . 130
Disabling and Reenabling the Cache Dbspace. . . . 131
Changing the Read-Only/Read-Write Status of a Cache Dbspace Dbfile. . . . 131
Monitoring the Cache Dbspace. . . . 132

7 Manage SAP IQ Tables and Views. . . . 136
7.1 Table Management. . . . 137
Creating a Table. . . . 137
Altering a Table. . . . 142

Dropping a Table. . . . 143
Creating Primary Keys. . . . 144
Creating Foreign Keys. . . . 144
Table Information in System Views. . . . 145
7.2 Table Partitions. . . . 145
Restrictions on Table Partitions. . . . 146
Range Partitions. . . . 147
Hash Partitions. . . . 149
Hash-Range Partitions. . . . 150
7.3 View Management. . . . 151
Creating Views. . . . 152
Guidelines for Using Views. . . . 153
Guidelines for Modifying Views. . . . 154
Permissions on Views. . . . 155
How to Delete Views. . . . 155
View Information in System Views. . . . 155

8 Index SAP IQ Columns. . . . 156
8.1 Overview of Indexes. . . . 157
8.2 Index Types Comparison. . . . 157
Containment (WD) Index. . . . 158
Compare (CMP) Index. . . . 160
Date (DATE) and Datetime (DTTM) Indexes. . . . 161
Fast Projection (FP) Index. . . . 166
High_Group (HG) Index. . . . 167
High_Non_Group (HNG) Index. . . . 169
TEXT Index. . . . 169
Time (TIME) Index. . . . 170
8.3 Criteria for Choosing Indexes. . . . 172
Number of Unique Values in the Index. . . . 172
Types of Queries. . . . 173
Indexes Recommended by Query Type. . . . 174
Indexes Recommended by Column Value. . . . 174
Disk Space Usage. . . . 175
Data Types in the Index. . . . 176
Index Type Combinations. . . . 177
8.4 Creating Indexes. . . . 177
Concurrent Column Indexing. . . . 178
Status Messages for Index Loading. . . . 178
Adding Column Indexes After Inserting Data. . . . 179
Executing Groups of CREATE INDEX Statements. . . . 179
8.5 Running the Index Advisor. . . . 179

8.6 Renaming Indexes. . . . 180
8.7 Viewing Indexes. . . . 180
Index Information in System Views. . . . 181
8.8 Removing Indexes. . . . 181
8.9 Retaining Indexes When Removing Foreign Key Constraints. . . . 182
8.10 Optimizing Performance for Joins. . . . 182

9 Enforce Data Integrity. . . . 184
9.1 Data Integrity Overview. . . . 184
How Data Can Become Invalid. . . . 185
Rules and Checks for Valid Data. . . . 185
Statements That Change Database Contents. . . . 186
Data Integrity Tools. . . . 186
Statements that Implement Integrity Constraints. . . . 187
9.2 Column Defaults Encourage Data Integrity. . . . 187
Supported Default Values. . . . 188
Default Value Restrictions. . . . 189
Creating Column Defaults. . . . 189
Changing Column Defaults. . . . 190
Deleting Column Defaults. . . . 190
Supported Column Default Values. . . . 190
Date, Time, and Timestamp Defaults. . . . 191
USER Defaults. . . . 192
The IDENTITY or AUTOINCREMENT Default. . . . 193
The NEWID Default. . . . 194
The NULL Default. . . . 194
String and Number Defaults. . . . 194
Constant Expression Defaults. . . . 194
9.3 Table and Column Constraints. . . . 195
UNIQUE Constraints on Columns or Tables. . . . 195
IQ UNIQUE Constraints on Columns. . . . 196
CHECK Conditions on Columns. . . . 196
CHECK Conditions on User-Defined Data Types. . . . 197
CHECK Conditions as a Constraint on a Table. . . . 198
CHECK Conditions on Tables. . . . 198
Removing Check Conditions from Tables. . . . 199
9.4 Entity and Referential Integrity. . . . 199
If a Client Application Breaches Entity Integrity. . . . 200
Referential Integrity. . . . 200
Concurrent Operations. . . . 205
Disabling Referential Integrity Checking. . . . 207
9.5 Integrity Rules in System Tables. . . . 207

9.6 Validating Catalog Store Indexes. . . . 208

10 Manage Transactions and Versioning. . . . 209

10.1 Transactions. . . . 210
Viewing Transaction Activity. . . . 211
Isolation Levels. . . . 211

10.2 Transaction Blocking. . . . 212
Enabling Connection Blocking. . . . 213
Disabling Connection Blocking. . . . 214
Setting the Blocking Timeout Threshold. . . . 214
Transaction Blocking Deadlocks. . . . 215

10.3 Versions. . . . 217
Table-Level Snapshot Versioning. . . . 218
Row-Level Snapshot Versioning. . . . 224
Viewing the Versioning Type of a Transaction. . . . 232
Temporary Table Versioning. . . . 233

10.4 Investigating Lock Contention Effects on Performance. . . . 233
(Deprecated) -contention. . . . 234

10.5 Checkpoints, Savepoints, and Transaction Rollback. . . . 235
Checkpoints. . . . 236
Savepoints Within Transactions. . . . 236
Transaction Rollback. . . . 238
System Recovery. . . . 239
How Transaction Information Aids Recovery. . . . 240
Concurrency for Backups. . . . 240

10.6 Cursors in Transactions. . . . 240
Cursors and Versioning. . . . 241
Cursor Sensitivity. . . . 242
Cursor Scrolling. . . . 242
Hold Cursors. . . . 242
Positioned Operations. . . . 243
Message Logging for Cursors. . . . 243

10.7 Remote Transactions. . . . 243
Remote Transaction Restrictions. . . . 243
Remote transaction management and restrictions. . . . 244

11 Create Procedures and Batches. . . . 245

11.1 Procedures. . . . 246
Creating Procedures. . . . 247
Altering Procedures. . . . 247
Calling Procedures. . . . 248
Deleting Procedures. . . . 248

Privileges to Execute Stored Procedures. . . . 248
Returning Procedure Results in Parameters. . . . 248
Returning Procedure Results in Result Sets. . . . 249
Displaying Procedure Information. . . . 249
Displaying Procedure Parameter Information. . . . 250
Cursors in Procedures. . . . 250
Using IQ UTILITIES to Create Stored Procedures. . . . 253
11.2 User-Defined Functions. . . . 256
Creating User-Defined Functions. . . . 256
Calling User-Defined Functions. . . . 257
Dropping a user-defined function (SQL). . . . 257
Permissions to Execute User-Defined Functions. . . . 258
Granting the ability to execute a user-defined function (SQL). . . . 258
11.3 Batches. . . . 259
11.4 Control statements. . . . 262
Compound statements. . . . 263
Declarations in compound statements. . . . 263
Atomic compound statements. . . . 264
11.5 Structure of Procedures. . . . 265
SQL Statements Allowed in Procedures. . . . 265
Parameter declaration for procedures. . . . 266
Ways to pass parameters to procedures. . . . 266
How to pass parameters to functions. . . . 267
11.6 Procedure Results. . . . 268
Returning a value using the RETURN statement. . . . 269
Ways to return results as procedure parameters. . . . 269
Information returned in result sets from procedures. . . . 271
Returning multiple result sets. . . . 273
Variable result sets for procedures. . . . 274
11.7 Error and warning handling. . . . 275
Default handling of errors. . . . 275
Error handling with ON EXCEPTION RESUME. . . . 277
Default handling of warnings. . . . 278
Exception handlers. . . . 279
Exception handling and nested compound statements. . . . 280
11.8 Transactions and savepoints in procedures 282
11.9 Hiding the contents of a procedure, function, trigger, event, or view. . . . 283
11.10 Statements allowed in procedures, triggers, events, and batches. . . . 285
SELECT statements used in batches. . . . 285
EXECUTE IMMEDIATE used in procedures, triggers, user-defined functions, and batches. . . . 286

12 Automate Tasks Using Schedules and Events. . . . 289

12.1 Task automation using schedules and events. 289
Events. 290
Schedules. 290
System events. 291
Event handlers. 294

13 Audit Database Events. 299
13.1 Database activity audits. 299
Configuring auditing (SQL). 300
Retrieving auditing information (dbtran utility). 301

14 Access Remote Data. 303
14.1 SAP IQ and Remote Data. 303
Characteristics of Open Client and jConnect connections. 304
Requirements for Accessing Remote Data. 306
Remote Servers. 327
External Logins. 338
Proxy tables. 339
Joins between remote tables. 343
Joins between tables from multiple local databases. 344
Native statements and remote servers. 345
Remote Procedure Calls (RPCs). 345
14.2 Internal Operations. 346
Query Parsing. 347
Query Normalization. 347
Internal operations performed on queries. 347
Complete passthrough of the statement. 348
Partial passthrough of the statement. 348
14.3 Remote Data Access Troubleshooting. 349
Troubleshooting: Features not supported for remote data. 350
Troubleshooting: Case sensitivity and remote data access. 350
Troubleshooting: Connectivity tests for remote data access. 351
Troubleshooting: Remote data access connections via ODBC. 351
Remote data access on multiplex servers. 351

15 OLAP. 352
15.1 About OLAP. 352
OLAP Benefits. 353
OLAP Evaluation. 354
15.2 GROUP BY Clause Extensions. 354
Group by ROLLUP and CUBE. 355
15.3 Analytical Functions. 367

Simple Aggregate Functions. . . . 368
Windowing. . . . 368
Numeric Functions. . . . 393
15.4 OLAP Rules and Restrictions. . . . 395
15.5 Additional OLAP Examples. . . . 396
Example: Window Functions in Queries. . . . 397
Example: Window with Multiple Functions. . . . 399
Example: Calculate Cumulative Sum. . . . 399
Example: Calculate Moving Average. . . . 400
Example: ORDER BY Results. . . . 400
Example: Multiple Aggregate Functions in a Query. . . . 401
Example: Window Frame Comparing ROWS and RANGE. . . . 401
Example: Window Frame Excludes Current Row. . . . 402
Example: Window Frame for RANGE 402
Example: Unbounded Preceding and Unbounded Following. . . . 403
Example: Default Window Frame for RANGE. . . . 404
15.6 BNF Grammar for OLAP Functions. . . . 405

16 Troubleshooting Hints. . . . 410
16.1 Online Resources. . . . 410
Community. . . . 410
Documentation. . . . 411
Video Catalog. . . . 411
Demos. . . . 411
Near-Line Storage Solution. . . . 411
SAP IQ and Hadoop. . . . 412
Whitepapers. . . . 412
16.2 Solutions for Specific Conditions. . . . 412
Decision Flow for Server Recovery and Database Repair. . . . 413
Server Operational Issues. . . . 414
Database Connection Issues. . . . 423
Interactive SQL (dbisql) Issues. . . . 425
Resource Issues. . . . 425
Processing Issues. . . . 434
Performance Issues. . . . 439
16.3 Troubleshooting Network Communications. . . . 440
Using Compatible Protocols. . . . 441
Using Current Drivers. . . . 441
Powering Down Your Computer Between Restarts. . . . 441
Diagnosing the Protocol Stack Layer by Layer. . . . 441
Testing a TCP/IP Protocol Stack. . . . 442
Diagnosing Wiring Problems. . . . 443

Checking Common Network Communications Problems. . . . 443
16.4 Diagnostic Tools. . . . 444
Restoring to a New Temporary File Topology. . . . 445
The sp_iqstatus Stored Procedure. . . . 446
Interpreting Notification Messages. . . . 448
The sp_iqcheckdb Stored Procedure. . . . 453
Checking Database and Server Startup Option Values. . . . 453
Finding the Currently Executing Statement. . . . 454
Logging Server Requests. . . . 454
Connection for Collecting Diagnostic Information. . . . 458
Diagnosing Communications Issues. . . . 459
16.5 Reporting Problems to Technical Support. . . . 459
Collecting Diagnostic Information Using getiqinfo. . . . 459
Information Collected by getiqinfo. . . . 461
Correlating Connection Information Between the .srvlog and .iqmsg Files. . . . 462
16.6 Checklist: Information for Technical Support. . . . 463

17 Connection and Communication Parameters Reference. . . . 465
17.1 Connection Parameters. . . . 465
AppInfo Connection Parameter. . . . 468
AutoPreCommit Connection Parameter. . . . 470
AutoStart (ASTART) Connection Parameter. . . . 471
AutoStop (ASTOP) Connection Parameter. . . . 472
CharSet (CS) Connection Parameter. . . . 473
CompressionThreshold (COMPTH) connection parameter. . . . 474
CommBufferSize (CBSize) Connection Parameter. . . . 475
CommBufferSpace (CBSpace) Connection Parameter. . . . 476
CommLinks (LINKS) Connection Parameter. . . . 477
ConnectionName (CON) Connection Parameter. . . . 479
ConnectionPool (CPOOL) Connection Parameter. . . . 480
DatabaseFile (DB) Connection Parameter. . . . 481
DatabaseName (DBN) Connection Parameter. . . . 482
DatabaseSwitches (DBS) Connection Parameter. . . . 484
DataSourceName (DSN) Connection Parameter. . . . 485
DBKEY Connection Parameter. . . . 486
DisableMultiRowFetch (DMRF) Connection Parameter. . . . 486
EngineName (ENG) Connection Parameter. . . . 487
EncryptedPassword (ENP) Connection Parameter. . . . 488
Encryption (ENC) connection parameter. . . . 489
Escape (ESCAPE) Connection Parameter. . . . 492
FileDataSourceName (FILEDSN) Connection Parameter. . . . 493
Idle (IDLE) Connection Parameter. . . . 494

Integrated (INT) Connection Parameter. 495
Language (LANG) Connection Parameter. 496
LazyClose (LCLOSE) Connection Parameter. 497
LivenessTimeout (LTO) Connection Parameter. 498
LogFile (LOG) Connection Parameter. 499
LogicalServer (LS) Connection Parameter. 500
LoginRedirection (REDIRECT) Connection Parameter. 501
NewPassword (NEWPWD) connection parameter. 502
NodeType Connection Parameter. 503
Password (PWD) Connection Parameter. 504
PrefetchBuffer (PBUF) Connection Parameter. 505
PrefetchRows (PROWS) Connection Parameter. 507
RetryConnectionTimeout (RetryConnTO) connection parameter. 508
ServerName Connection Parameter. 509
StartLine (START) Connection Parameter. 509
Unconditional (UNC) Connection Parameter. 510
Userid (UID) Connection Parameter. 511
17.2 Network Communications Parameters. 512
Broadcast (BCAST) Communication Parameter. 514
BroadcastListener (BLISTENER) Communication Parameter. 515
ClientPort (CPort) Communication Parameter. 516
DatabaseName (DBN) Communication Parameter. 517
DoBroadcast (DBROAD) Communication Parameter. 518
Host (IP) Communication Parameter. 520
Identity Communication Parameter. 521
Identity_Password Communication Parameter. 523
LDAP Communication Parameter. 524
LocalOnly (LOCAL) Communication Parameter. 525
LogFile (LOG) Communication Parameter. 526
LogFormat (LF) Communication Parameter. 526
LogMaxSize (LSIZE) protocol option. 527
LogOptions (LOPT) Communication Parameter. 528
MaxConnections (MAXCONN) Communication Parameter. 530
MaxRequestSize (MAXSIZE) protocol option. 530
MyIP (ME) Communication Parameter. 531
PreFetchOnOpen Communication Parameter. 533
ReceiveBufferSize (RCVBUFSZ) Communication Parameter. 534
SendBufferSize (SNDBUFSZ) Communication Parameter. 534
ServerPort (PORT) Communication Parameter. 535
Sessions Communication Parameter. 537
TDS Communication Parameter. 537

Timeout (TO) Communication Parameter. 538
VerifyServerName (VERIFY) Communication Parameter. 540

1 SAP IQ Administration: Database

Database setup and administration.

2 Database Administration Workflow

Creating an SAP IQ database is part of a larger setup process that begins with installation and ends when your database is available to users.

Procedure

1. Install and configure SAP IQ.

   Install both the client and server environments. See the *SAP IQ Installation and Update Guide* for your platform.
2. Start and connect to a database server.

   Enable database control and open a communication channel to the database. Use the `start_iq` utility, SAP IQ Cockpit, the Windows Start menu, or a configuration file.
3. Create a database.

   Create both the IQ store and the catalog store. You may use SAP IQ Cockpit, the SQL statement `CREATE DATABASE`, or the `iqinit` utility.
4. Create the tables in your database.

   Use the `CREATE TABLE` statement or the SAP IQ Cockpit Create Table wizard.
5. Create indexes for the tables.

   Use the `CREATE INDEX` statement or the SAP IQ Cockpit Create Index wizard. You can also create certain indexes automatically when you create your tables.
6. Load data into the tables.

   Use the `LOAD TABLE` statement to bulk-load data from files, or use the `INSERT` statement to extract rows of data from an existing database. See *Bulk Loads with the LOAD TABLE Statement* in *Administration: Load Management*.

3 Run SAP IQ Database Servers

Each SAP IQ database runs on a server. Depending on your platform and administration tool, SAP IQ offers multiple ways to start and stop a server.

In this section:

Starting Servers [page 16]
Use commands or graphical product interfaces to start servers.

Command Line Options [page 20]
Command line options define your SAP IQ environment.

Multiple Servers on a System [page 27]
In a production environment, it would be unusual to have more than one server running on the same system, and SAP strongly recommends against it. In a development environment, however, this situation can occur.

Configuration Files [page 27]
You can store sets of command line switches in configuration files.

Server Activity Logs [page 29]
Using commands appropriate for your platform, you can direct SAP IQ to capture server activity in a log file.

Stopping Servers [page 32]
Different ways to stop a server.

Starting and Stopping Databases [page 36]
You can start databases when you start the server, or after the server is running.

Starting the iqdemo Database [page 37]
Use the script provided at installation to create the `iqdemo` database and the configuration file to start it.

3.1 Starting Servers

Use commands or graphical product interfaces to start servers.

Server Startup Methods for All Platforms

Choose the startup method for your interface to SAP IQ.

To Start the Server From the ...	Do This ...
Server startup utility	Enter a `start_iq` command on the command line. Syntax: `start_iq [ <server-options> ] [ <database-file> [ <database-options> ]<, ...>]` If you enter several command line options, you must enter them all on one line (without carriage returns). For ease of use, specify the database name when you start the server. This starts the database and server together. Omit the database name to start the server without starting a database. If you omit the database name, you must name the server explicitly using the `-n` server switch. Use this method when you create or restore a database, or when you only want to control starting and stopping the server, leaving database use to client software.
Configuration file	Start the server and the `iqdemo` database with an SAP-provided configuration file.
Server startup command	Place a server startup command in a shortcut or desktop icon.
ODBC data source	Include a server start line in the data source.
Utility command	Include a server start line in a utility command.
Interactive SQL	Issue a SQL command from Interactive SQL to start an additional server.
SAP IQ Cockpit	To start in SAP IQ Cockpit, see the SAP IQ Cockpit for SAP IQ or at https://help.sap.com/viewer/p/SAP_IQ.

Windows Server Startup Methods

There are several ways start a Windows server.

To Start the Server From...	Do This...
Command line	Run the `stop_iq` or `dbstop` command.
Desktop	Right-click the **server icon** in the system tray, and choose `Start server name`.
Windows Start menu	Click **Start** > **All Programs** > **SAP** > **SAP IQ 16.0** > **Start SAP IQ Demo Database**.
Service Manager	If you started the server as a Windows service: 1. On the **Control Panel**, choose **Administrative Tools** > **Services**. 2. Choose the SAP IQ service, then click `Start the service`.

In this section:

The Server Startup Utility [page 18]
The startup utility, `start_iq`, runs on all platforms, and ensures that required parameters are set correctly, with a few documented exceptions.

Starting the Server Using the Startup Utility [page 19]
The `start_iq` command starts the named server as a background process, starts the named database if you specify it, and sets all required startup options.

Starting a Server from Interactive SQL [page 19]
If you are already connected to a running database server, you can start a new server from Interactive SQL.

Automatic Server Startup [page 20]
Use SAP IQ Service Manager to define a service that starts an SAP IQ server.

3.1.1 The Server Startup Utility

The startup utility, `start_iq`, runs on all platforms, and ensures that required parameters are set correctly, with a few documented exceptions.

In examples that include several command line options, they are, for clarity, shown on separate lines, as they could be written in a configuration file. If you enter them directly on a command line, you must enter them all on one line (that is, without any carriage returns).

You can choose from many command line switches to specify such features as permissions required to start a database or stop the server, and the network protocols to use. Command line options are one way to tune SAP IQ behavior and performance.

Startup of Database with Database Server

For ease of use, start the database and server together, by specifying the database name when you start the server.

By default, the server takes its name from the database name, or you can specify a different name for the server.

Startup of Database Server Without Database Startup

To start the server without starting any database, omit the database file from the `start_iq` command and specify a server name.

If you omit the database name, you must name the server explicitly using the -n server switch. Use this method when you create or restore a database, or when you only want to control starting and stopping the server, leaving database use to client software.

3.1.2 Starting the Server Using the Startup Utility

The `start_iq` command starts the named server as a background process, starts the named database if you specify it, and sets all required startup options.

Procedure

1. Change to a writable directory.
2. At the system prompt, enter:

   ```
   start_iq <servername> [ <database> ]
   ```

   If you do not specify the database, you must use `-n <server name>` or the server does not start. In this example, the server starts on the default port, 2638.

3.1.3 Starting a Server from Interactive SQL

If you are already connected to a running database server, you can start a new server from Interactive SQL.

Prerequisites

This method is not recommended for most situations. If you use it, ensure the following:

- Start the server on the intended system
- Include appropriate server parameters in the command
- Set environment variables appropriately on the system where the server will start

Context

Procedure

Use the `START ENGINE` command to start a named server from Interactive SQL.

The following Interactive SQL command, entered on one line, starts a database server, names it `jill_newserv`, and specifies the network connection, number of connections, and catalog page size:

```
START ENGINE AS jill_newserv
```

```
STARTLINE 'start_iq -x tcpip(port=5678) -gm 10 -gp 4096'
```

3.1.4 Automatic Server Startup

Use SAP IQ Service Manager to define a service that starts an SAP IQ server.

You can then configure the service to automatically start the server whenever the host is started. The service may start either non-multiplex or multiplex servers.

When run as a Windows service, an SAP IQ server continues running even after you log off the machine.

For details of this and other Windows-specific features, see the *SAP IQ Installation and Update Guide*.

3.2 Command Line Options

Command line options define your SAP IQ environment.

For a complete list of command line options and full reference information, see *start_iq Database Server Startup Utility* in *SAP IQ Utility Reference*.

Some of the values you can set with command line options can also be changed with the `SET OPTION` command.

> **i** Note
>
> For clarity, examples throughout SAP IQ documentation show multiple command line options on separate lines as you can write them in a configuration file.
>
> If you enter them directly on a command line, you must enter them all on one line.

In this section:

Displaying Command Line Options [page 21]
Command line switches are case sensitive.

Required Command Line Options [page 21]
Not all command line switches are optional.

Server and Database Names [page 21]
Use the `-n` command line switch as a server switch (to name the server). This prevents you from unintentionally connecting to the wrong server.

Performance Options [page 23]
Server startup options control network performance, timing, overhead for managing transactions, and buffers for data reads.

Privilege Options [page 25]
Server startup options set the privilege level of some database and server functions.

Client/Server Environment Options [page 25]

Server startup options control communication protocols and connection timeouts.

Catalog Page Size Option [page 26]
The database server cache is arranged in pages, which are fixed-size areas of memory. Because the server uses a single cache for the catalog store until it is shut down, all catalog pages must have the same size.

3.2.1 Displaying Command Line Options

Command line switches are case sensitive.

Procedure

To display all of the available command line switches, enter this command at the operating system prompt:

```
start_iq -h
```

3.2.2 Required Command Line Options

Not all command line switches are optional.

While most of the command line switches described in *SAP IQ Utility Reference* > *start_iq Database Server Startup Utility* are optional, you must specify the `-n` switch to run SAP IQ effectively.

> **i** Note
>
> You should use `-c 48M` on all 64-bit platforms.

If you use TCP/IP to connect to the server, include network connection parameters as well. If you start the server without the parameter `-x 'tcpip(port=nnnn)'`, the server uses the default TCP/IP port number, 2638. If you specify a port number that is already in use, the server fails to start.

3.2.3 Server and Database Names

Use the `-n` command line switch as a server switch (to name the server). This prevents you from unintentionally connecting to the wrong server.

The server and database names are among the connection parameters that client applications can use when connecting to a database. On Windows, the server name appears on the desktop icon and on the title bar of the server window.

> **Note**
>
> While you can start more than one database, you should run only one database per SAP IQ server. If you must run multiple databases, start each one on a separate IQ database server, and on a different port.

Server Names

Name a server by supplying a -n switch before the first database file. The rest of the parameters are added from the `default.cfg` file.

For example, this command starts a server named `Cambridge_iqdemo` and the `iqdemo` database on that server:

```
start_iq -n Cambridge_iqdemo iqdemo.db
```

Each server name must be unique across the local area network (domain). This prevents you from unintentionally connecting to the wrong server. The host name and port number combination does not uniquely identify the server. Appending a unique identifier to the server name is a useful convention. Assigning unique server names is especially important in a multiuser, networked environment where shared memory is used for local database connections. This convention ensures that all users can connect to the correct database, even when other databases with the same name have been started on other host systems.

To allow SAP IQ to locate the server regardless of the character set in use, include only seven-bit ASCII (lower page) characters in the server name.

Specifying a server name lets you start a database server with no database loaded. This command starts a server named `Galt` with no database loaded:

```
start_iq -n Galt -gm 10 -gp 4096
```

> **Note**
>
> Although you can start a server by relying on the default server name, it is a better practice to include both the server name and the database name, and to make the two names different. This approach helps users distinguish between the server and the databases running on it. You must specify the server name to start the server without starting a specific database.

Database Names

You can name databases by supplying a `-n` switch following the database file.

For example, this command line starts a database and names it:

```
start_iq -n MyServer mydb.db -n MyDB
```

Naming a database lets you use a nickname.

Naming Conventions

If no server name is provided, the first database started provides the default server name.

Server names and database names are case-insensitive on Windows, and case-sensitive on UNIX.

Adopt a set of naming conventions for your servers and databases, as well as for all other database objects, which includes a case specification. Enforcing naming conventions can prevent problems for users.

Avoid hyphenated names or reserved words for database names, user identifiers, or server names, even enclosed in quotation marks.

For example, the following are not allowed:

- `grant`
- `june-1999-prospects`
- `"foreign"`

See *Reserved Words* in *SAP IQ SQL Reference*.

3.2.4 Performance Options

Server startup options control network performance, timing, overhead for managing transactions, and buffers for data reads.

Most of the performance command line options control resources for operations on the IQ store, which can have a major impact on performance. Options that affect only the resources available for operations on the catalog store may have a minor impact on overall performance.

Name	Description
`-c`	Sets initial memory reserved for caching catalog store pages and other server information. Usage: `-c <cache-size> [k | m | g | p]`
`-gb`	Sets server priority class. Usage: `-gb { idle | normal | high | maximum] ... | level ...}`
`-gc`	Sets maximum interval between checkpoints. Usage: `-gc <minutes>`
`-iqgovern`	Sets number of concurrent queries allowed by the server. Helps SAP IQ optimize paging of buffer data out to disk and avoid overcommitting memory. Usage: `-iqgovern <num>`

Name	Description
-iqlm	Specifies maximum amount of memory in MB that SAP IQ can dynamically request from the operating system for temporary use. Usage: `-iqlm <size>`
-iqlrt	Controls number of threads that SAP IQ uses during point-in-time recovery operations. Usage: `-iqlrt <size>`
-iqmc	Specifies main buffer cache size in MB. Usage: `-iqmc <size>`
-iqmt	Specifies the number of SAP IQ threads to create. `-iqmt <num>`
-iqpartition	Sets the number of partitions in the IQ main and temporary main buffer caches. Usage: `-iqpartition <num>`
-iqtc	Specifies IQ temporary cache size in MB. Usage: `-iqtc <size>`
-iqwmem	Creates a pool of “wired” memory on HP and Sun UNIX systems, in MB. Usage: `-iqwmem <size>`
-iqnumbercpus	Specifies the number of CPUs available, for resource planning purposes. Usage: `-iqnumbercpus <num>`
-p	Sets the maximum size of communication packets. Usage: `-p <integer>`

3.2.5 Privilege Options

Server startup options set the privilege level of some database and server functions.

Name	Description
`-gd`	Sets the privileges for starting or stopping databases on a running server. Usage: `-gd { DBA | all | none } ...` See *-gd database server option*.
`-gk`	Sets the privileges for starting or stopping databases on a running server. Usage: `-gk { DBA | all | none } ...` See *-gk database server option*.
`-gl`	Sets the permission required to load data using LOAD TABLE. Usage: `-gl <level>` See *-gl database server option*.
`-gu`	Sets the privileges for running database file administration statements, such as those for creating or dropping databases. Usage: `-gu { all | none | DBA | utility_db } ...` See *-gu database server option*.

3.2.6 Client/Server Environment Options

Server startup options control communication protocols and connection timeouts.

Name	Description
`-ti`	Disconnects inactive connections. Usage: `-ti minutes ...` See *-ti database server option*.
`-tl`	Sets the period at which to send liveness packets. Usage: `-tl seconds ...` See *-tl database server option*.

Name	Description
`-x`	Specifies server-side network communications protocols. Usage: `-x <list>` See *-x database server option*.

> **Note**
>
> Users who are running a client and server on the same machine do not experience a liveness timeout.

The `start_iq` default for `-tl` is 4400 (about 72 hours), which lets you start long runs at the beginning of a weekend, for example, and ensure that any interim results are not rolled back.

3.2.7 Catalog Page Size Option

The database server cache is arranged in pages, which are fixed-size areas of memory. Because the server uses a single cache for the catalog store until it is shut down, all catalog pages must have the same size.

A catalog file is also arranged in pages, of size 4096, 8192, 16384, or 32768 bytes. Every database page must fit into a cache page.

Use the `-gp` option to set the catalog page size explicitly. To maximize the number of columns per table that SAP IQ can support, set `-gp` to the maximum size, 32768. For example:

```
start_iq -n MyServer -gp 32768
```

By default, the server page size is the same as the largest page size of the databases on the command line. The `-gp` option overrides this default. Once the server starts, you cannot load a database with a larger catalog page size than the server. Unless you specify `-gp`, you cannot load a database file with a catalog page size larger than the databases started on the command line.

If you use larger page sizes, increase your cache size. A cache of the same size accommodates only a fraction of the number of the larger pages, leaving less flexibility in arranging the space.

> **Note**
>
> The `-gp` option and the page sizes listed here apply only to the catalog store. You set the page size for the IQ store in the IQ PAGE SIZE parameter of the `CREATE DATABASE` command.

3.3 Multiple Servers on a System

In a production environment, it would be unusual to have more than one server running on the same system, and SAP strongly recommends against it. In a development environment, however, this situation can occur.

If you run more than one server or client on the same UNIX machine, and shared memory is enabled, you must take certain precautions to prevent users from connecting to the wrong server.

When attempting to start a server, you may see:

```
DBSPAWN ERROR -96 -- database engine already running
```

This error indicates that the startup process is finding the shared memory segment of a server started earlier, and cannot create a shared memory segment. This error may occur when either an SAP IQ or SAP SQL Anywhere server is running. (Interactive SQL also connects to an earlier server if its shared memory port is visible, even if you intended for it to connect to a server started later.) You can avoid the error if you run only one server per system, either SAP IQ or SAP SQL Anywhere.

Ways to Avoid Shared Memory Conflicts

- Create a temporary directory that is dedicated to each server. Make sure that each client uses the same temporary directory as its server by setting the IQTMP16 environment variable explicitly on both systems.
- For each server, create a data source name in the `.odbc.ini` file (on UNIX) and provide detailed connection information.
- Use connection strings that specify explicit parameters instead of relying on defaults.
- Confirm connections by issuing:

  ```
  SELECT "database name is" = db_name(), "servername_is" = @@servername
  ```

If you run multiple servers per system, make sure that each server has a unique:

- Name, specified with the `-n` parameter on startup.
- Port number, specified with the `-x` parameter.

3.4 Configuration Files

You can store sets of command line switches in configuration files.

If you use an extensive set of command line switches, you can store them in a configuration file, and invoke that file in a server command line. Specify switches in the configuration file in the same way as on the command line, except in the file, you can enter switches on multiple lines.

In this section:

Configuration File List [page 28]

SAP IQ provides configuration files that you can use as templates.

Examples of Configuration Files [page 28]
You can use the predefined configuration files as templates to create your own.

Default Configuration File [page 29]
The default configuration file (`default.cfg`) contains all of the required startup switches.

Configuration File for the Demo Database [page 29]
The `iqdemo.cfg` file, which you use to start the demo database, sets startup switches to the recommended defaults.

3.4.1 Configuration File List

SAP IQ provides configuration files that you can use as templates.

File Name	Location	Use
`default.cfg`	(UNIX) `$IQDIR16/scripts` (Windows) `%IQDIR16%\scripts`	Generic configuration file. This file is used for default options for `start_iq` and multiplex startup. SAP IQ copies `default.cfg` into each new database directory and renames it `params.cfg`. Any changes that you make to `default.cfg` (in the `scripts` directory) are inherited by all databases that are created after the file is changed.
`iqdemo.cfg`	(UNIX) `$IQDIR16/data/demo` (Windows) `%IQDIR16%\data\demo`	Starts the demo database, setting startup switches to the recommended defaults.

3.4.2 Examples of Configuration Files

You can use the predefined configuration files as templates to create your own.

For example, the following configuration file starts the database `mydb.db` on the database server named `Elora`, with a 32 MB cache and a 20-minute checkpoint interval. It allows anyone to start or stop databases and load data, limits user connections to 10, has a catalog page size of 4096 bytes, has a default client connection timeout of 72 hours, uses TCP/IP as a network protocol, and has a specified port number of 1870:

```
-n Elora -c 32m -gc 20
-gd all -gl all -gm 10 -gp 4096 -ti 4400 -x tcpip(port=1870) path\mydb.db
```

You can execute all these command line options using:

```
start_iq @mydb.cfg
```

> **Note**
>
> When you stop the server with the `DBSTOP` command, you must specify the same parameters as when you started the server; if you use a configuration file to start the server ensures that you can find these parameters when you need them to stop the server.

3.4.3 Default Configuration File

The default configuration file (`default.cfg`) contains all of the required startup switches.

The `default.cfg` file:

- Provides the source for the `params.cfg` file used by the UNIX `start_iq` command
- Starts servers for Windows services

To override any switch except for `-n <servername>` in configuration files, specify new switches on the `start_iq` command line.

3.4.4 Configuration File for the Demo Database

The `iqdemo.cfg` file, which you use to start the demo database, sets startup switches to the recommended defaults.

You can automatically create this file when you create the demo database using scripts provided by the SAP IQ.

3.5 Server Activity Logs

Using commands appropriate for your platform, you can direct SAP IQ to capture server activity in a log file.

Server Startup Messages

When you start an SAP IQ server, a series of messages appears in the server log window. The exact set of messages you see depends on your platform and licensed options. This example is from a Solaris system:

```
Starting server <servername>_iqdemo on mo-abaa2973e at port 2638 (01/18 19:12:59)
Run Directory       : /SAP_IQ/IQ-16_1/demo
Server Executable   : /SAP_IQ/IQ-16_11/bin64/iqsrv16
Server Output Log   : /SAP_IQ/IQ-16_1/logfiles/<servername>_iqdemo.0009.srvlog
Server Version      : 16.1.010.844/Mainline
17.0.7/Linux 2.6.18-308.el5
Open Client Version : N/A
User Parameters     : '@iqdemo.cfg' 'iqdemo.db'
Default Parameters  : -ti 4400 -gn 25
```

```
I. 01/18 19:13:00. SAP IQ
I. 01/18 19:13:00. Version 16.0
I. 01/18 19:13:00. (64bit mode)
I. 01/18 19:13:00. Copyright 1992-2016 by SAP AG or an SAP affiliate company.
All rights reserved
I. 01/18 19:13:00. Copyright (c) 2016 SAP SE or an SAP affiliate company.
I. 01/18 19:13:00. All rights reserved.
I. 01/18 19:13:00. Use of this software is governed by the SAP Software Use
Rights Agreement.
I. 01/18 19:13:00. Refer to http://global.sap.com/corporate-en/our-company/
agreements/index.epx.
I. 01/18 19:13:00.
I. 01/18 19:13:00. Processors detected: 4 logical processor(s) on 4 core(s) on 4
physical processor(s)
I. 01/18 19:13:00. This server is licensed to use: all logical processors in the
system
I. 01/18 19:13:00. Processors in use by server: 4 logical processor(s) on 4
core(s) on 4 physical processor(s)
I. 01/18 19:13:00. Running Linux 3.0.101-0.47.71-default #1 SMP Thu Nov 12
12:22:22 UTC 2015 (b5b212e) on X86_64
I. 01/18 19:13:00. Server built for X86_64 processor architecture
I. 01/18 19:13:00. 49720K of memory used for caching
I. 01/18 19:13:00. Minimum cache size: 49728K, maximum cache size: 262144K
I. 01/18 19:13:00. Using a maximum page size of 4096 bytes
I. 01/18 19:13:00. Multiprogramming level: 25
I. 01/18 19:13:00. Automatic tuning of multiprogramming level is disabled
I. 01/18 19:13:00. Starting database "iqdemo" (/SAP_IQ/IQ-16_1/demo/iqdemo.db)
at Wed Jan 18 2017 19:13
I. 01/18 19:13:01. Transaction log: iqdemo.log
I. 01/18 19:13:01. Starting checkpoint of "iqdemo" (iqdemo.db) at Wed Jan 18
2017 19:13
I. 01/18 19:13:01. Finished checkpoint of "iqdemo" (iqdemo.db) at Wed Jan 18
2017 19:13
I. 01/18 19:13:01. Update previous log GUID to: a4cbbeae-
ddb1-11e6-8000-8f2d80cbf621
Update current log GUID to :21757fb2-ddb2-11e6-8000-b2847091cb49
I. 01/18 19:13:01. Database "iqdemo" (iqdemo.db) started at Wed Jan 18 2017 19:13
I. 01/18 19:13:01. IQ Server mo-abaa2973e_iqdemo.
I. 01/18 19:13:01. Starting checkpoint of "iqdemo" (iqdemo.db) at Wed Jan 18
2017 19:13
I. 01/18 19:13:01. Finished checkpoint of "iqdemo" (iqdemo.db) at Wed Jan 18
2017 19:13
I. 01/18 19:13:01. Database server started at Wed Jan 18 2017 19:13
I. 01/18 19:13:01. Trying to start SharedMemory link ...
I. 01/18 19:13:01.     SharedMemory link started successfully
I. 01/18 19:13:01. Trying to start TCPIP link ...
I. 01/18 19:13:01. Starting on port 2638
I. 01/18 19:13:06.     TCPIP link started successfully
I. 01/18 19:13:06. Now accepting requests
```

The start_iq Log File

When you start a server with the `start_iq` utility, server activity is logged in an ASCII text file placed in the directory defined by `$IQLOGDIR16` (for UNIX) or `%IQLOGDIR16%` (for Windows). This file contains the standard output from the server and the server status.

The log file name has this format:

```
your_server_name.nnnn.srvlog
```

Each time you start the server, the number is incremented. For example, your directory may look like this:

```
demo.0001.srvlog   demo.0002.srvlog
testdemo.0001.srvlog
```

For information about your most recent session, choose the log with the largest number for the desired server. Issue a `tail -f` command to view the log contents. For example:

```
% tail -f demo.0002.srvlog
```

If you do not define `$IQLOGDIR16` directory, then on UNIX, the log is written to `$IQDIR16/logfiles/` directory, and on Windows to the `%IQLOGDIR16%` directory defined by the SAP IQ installation.

When you run `start_iq`, specify the `-z` option to enhance the log file with additional information about connections, which may help new users or those troubleshooting connection problems.

On UNIX systems, there are two ways to check if a server is running.

- Log in to the machine where the server was started, and issue:

  ```
  ps -eaf | grep iqsrv
  ```

  This output differs slightly across UNIX platforms. For IBM AIX, the columns are:

  ```
  UID   PID  PPID   C    STIME TTY         TIME CMD
  ```

  For example:

  ```
  jones  422034      1   0 17:47:36      -  0:04
  /ibm64srv/users/mydatabase/iq160/IQ-16_1/bin64/
  iqsrv16
  @iqdemo.cfg iqdemo.db -ti 4400 -gn 25 -o
  /ibm64srv/users/mydatabase/iq160/IQ-16_1/logfiles/ibm64srv_iqdemo.0003.srvlog
  -hn 7
  ```

- Use the `stop_iq` utility, described in the following section, which displays all SAP IQ processes that are running.

On Windows systems, look in the system tray for one or more SAP IQ icons. Place the cursor over each icon and read the server name.

In this section:

Naming the Server Log File [page 32]
The server log name defaults to `server.nnnn.srvlog`.

UNIX Log Files [page 32]
On UNIX platforms, an additional log file captures operating system output, including stdout and stderr output.

3.5.1 Naming the Server Log File

The server log name defaults to `server.nnnn.srvlog`.

Procedure

Use the `-o` switch on the `start_iq` command to change the server log file name.

This example starts the server, and saves output to a file named `results` in the directory where the server was startedd:

```
start_iq -n imyserver -o results
```

You can also use the `-o` switch to specify the full path to the log file.

3.5.2 UNIX Log Files

On UNIX platforms, an additional log file captures operating system output, including stdout and stderr output.

The file name has this format:

```
your_server_name.####.stderr
```

For unexpected exceptions, SAP IQ writes a stack trace file. On UNIX systems, the name of the file that contains stack trace information has this format:

```
stktrc-YYYYMMDD-HHNNSS_#.iq
```

3.6 Stopping Servers

Different ways to stop a server.

Prerequisites

SERVER OPERATOR system privilege is the default permission level required to stop a server, but you can use the `-gk` startup option to change the default to ALL or NONE. Setting the permission level to ALL lets all users stop the server; setting the permission level to NONE means that no one can stop the server. In a production environment, only the Server Operator is allowed to stop the database server.

Do not shut down a server that is still connected to a client, or you will lose uncommitted transactions. Disconnect or close all the clients before shutdown.

UNIX Server Shutdown Commands

Several commands shut down the database server:

Command	Description
`stop_iq`	Shuts down a server completely without regard for users, connections, or load process status. Syntax: `stop_iq [ options ]`
`dbstop`	Stops a server or database. Additional options let you stop a server, even if there are active connections. Syntax: `dbstop [ options ] <server-name>` To use `dbstop` to stop a server, you specify a `<server-name>`, as well as any connection parameters you specified when you started the server.
`iqsrv16`	Stops a server when typed in the window where the server was started, if you have not redirected input to another device: `q`
`STOP ENGINE`	Stops a database server. Syntax: `STOP ENGINE engine-name [ UNCONDITIONALLY ]` The UNCONDITIONALLY argument stops a database server, even if there are client connections to the server.

Windows Server Shutdown Methods

There are several methods to shut down or stop a server:

- Command line – Run the `stop_iq` or `dbstop` command.
- Desktop – Right-click the **server icon** in the system tray, and choose `Shutdown server name`.
- Service Manager – If you started the server as a Windows service:
  1. On the **Control Panel**, choose **Administrative Tools** **Services**.
  2. Choose the SAP IQ service, then click **Stop the service**.
- Interactive SQL – Run the `stop engine` command to stop a named database server.

Stopping Servers in Cron or At Jobs on UNIX

To use `stop_iq` in a `cron` or `at` job, specify the full path name to the `stop_iq` executable and the appropriate `-stop` option.

Setting `-stop one` shuts down a single server, when exactly one running server was started by the user ID that starts the `cron` or `at` job. This prevents accidentally shutting down the wrong server if several are running:

```
stop_iq -stop one
```

Setting `-stop all` shuts down all servers that were started by the user ID that starts the `cron` or `at` job:

```
stop_iq -stop all
```

Examples

Example 1 – Stop a Server Using stop_iq

The following example uses the `stop_iq` utility in a UNIX operating system command line to shut down an SAP IQ server and close all user connections to it.

When you issue the `stop_iq` command, SAP IQ lists all servers owned by other users, followed by the servers you own. It then asks if you want to stop your server. For example:

```
% stop_iq
```

```
Checking system for IQ 16 Servers ...
The following 2 server(s) are owned by other users.
##     Owner   PID   Started   CPU_Time  Additional Information
--     -----  -----  --------  --------  ----------------------
      handari  19895  15:43:44   183:38
start_iq @iqdemo.cfg iqdemo.db -gn 105 -o /server1/users/surya/IQ-16_1/logfiles/
surya_ibm2.001.srvlog -hn 8       pamela    409802  18:05:02    0:05
SVR:ibm1_iqdemo2 DB:iqdemo
PORT:2678/ibm1/users/IQ/iq16/IQ-16_1/bin64/iqsrv16 @iqdemo.cfg
iqdemo.db -ti 4400 -gn 25 -o /ibm1/users/mydatabase/iq16/IQ-16_1/logfiles/
ibm64qa_iq
The following 1 server(s) are owned by 'kermit'
##     Owner    PID   Started   CPU_Time  Additional Information
-- ---------   -----  --------  --------  ----------------------
1:     kermit  422034  15:11:37     0:07  SVR:myserver_iqdemo
DB:iqdemo PORT:2638 /myserver/users/mydatabase/iq16/IQ-16_1/bin64/iqsrv16
@iqdemo.cfg iqdemo.db -ti 4400 -gn 25 -o /myserver/users/mydatabase/iq16/IQ-16_1/
logfiles/myserver_iq
start_iq -c 32m -gd all -gm 10 -gn 25 -gp 4096 -ti 4400 -tl 300 @iqdemo.cfg
--
    Please note that 'stop_iq' will shut down a server completely
    without regard for users connections or load processes status.
    For more control, use the 'dbstop' utility, which has options
    that control stopping servers based on active connections.
Do you want to stop the server displayed above <Y/N>?
```

To shut down the server, type **Y** (yes). You see messages similar to:

```
Shutting down server (422034) ...
Checkpointing server (422034) ...
Server shutdown.
```

To leave the server running, type **N** (no). You return to the system prompt and the server does not shut down.

If no running servers were started by your user ID, SAP IQ displays information about servers run by other users, then a message similar to:

```
There are no servers owned by 'kermit'
```

Example 2 – Stopping a Server From Interactive SQL

The following example stops a server from Interactive SQL:

```
STOP ENGINE Ottawa UNCONDITIONALLY
```

The optional keyword `UNCONDITIONALLY` specifies that the database server will be stopped even if there are connections to it.

> **i** Note
>
> You can stop a server from Interactive SQL if you are connected as DBA to one of the databases running on that server (including the `utility_db` database), or if the server was started with the `-gk ALL` option.

In this section:

Operating System Session Shutdown [page 35]
Always stop the database server explicitly before closing the operating system session.

When to Stop and Restart the Server [page 35]
In some situations, the server may need to be stopped and restarted.

3.6.1 Operating System Session Shutdown

Always stop the database server explicitly before closing the operating system session.

If you close an operating system session where a database server is running, or if you use an operating system command (other than `stop_iq`) to stop the database server, the server shuts down, but not cleanly. Next time the database is loaded, recovery happens automatically.

> **i** Note
>
> An example of a command that does not stop a server cleanly is stopping the process in the Windows Task Manager Processes window.

3.6.2 When to Stop and Restart the Server

In some situations, the server may need to be stopped and restarted.

For example:

- To install a new version of SAP IQ
- To reset some server command line options

- To cause a small number of server-wide database options to take effect
- Before closing the operating system session

3.7 Starting and Stopping Databases

You can start databases when you start the server, or after the server is running.

Run only one database per server, especially in a production environment.

There are several ways to start a database on a running server:

- To start a database from Interactive SQL or Embedded SQL, use the `START DATABASE` statement.
- To start and connect to a database from Interactive SQL, use a data source that specifies the database file.
- To start and connect to a database when you start Interactive SQL from a system command prompt, include the connection parameter `"DBF=db-file"`.
- To start an embedded database while connected to a server, connect to a database using a DBF parameter that specifies a database file for a new connection. The database file is loaded onto the current server.

In this section:

Database Startup Guidelines [page 36]
There are several issues to consider regarding database startup.

Stopping Databases [page 37]
Different ways to stop a database.

3.7.1 Database Startup Guidelines

There are several issues to consider regarding database startup.

File Access

For a database to start, all files of IQ_SYSTEM_MAIN, all files of IQ_SYSTEM_TEMP, and the catalog file SYSTEM must be available. You can start a database while skipping dbspaces that cannot be fully opened. If any writable files of IQ main store dbspaces other than IQ_SYSTEM_MAIN or any catalog dbspace files other than SYSTEM cannot be opened on server startup, SAP IQ logs an error and marks the dbspace dynamically offline (marked offline in memory, as opposed to marking it offline in the catalog). If there are any files of IQ_SYSTEM_TEMP that cannot be opened, the database does not start unless you use the `-iqnotemp` startup parameter.

SAP IQ checks the consistency of the commit_id in each dbspace file header against the value in the system tables `ISYSDBFILE` and `ISYSIQDBSPACE` and marks any file or dbspace that does not match offline as above.

A dbspace that has been marked offline at start time may be brought online via the `ALTER DBSPACE ONLINE` statement, assuming that the problem has been corrected and the dbspace can be opened. To correct path

problems, you can correct the path of the dbspace file using `ALTER DBSPACE <dbspace name>`, `ALTER FILE< logical filename>`, and `RENAME PATH <new pathname>`.

A table object that resides in an offline dbspace is unavailable. Any DDL or DML request except `ALTER DBSPACE ONLINE`, to any table object in an offline dbspace generates an error. After you make a dbspace offline, there may still be data pages in the buffer cache. In the case of a very small table, the entire table may be in memory in the buffer cache and temporarily available, even if the dbspace is offline.

Page Size Functional Restrictions

The server holds database information in memory using fixed-size pages. Once a server has been started, you cannot load a database that has a catalog or IQ page size larger than the server. For this reason, always set the catalog page size to its maximum value, 32768 bytes, with the `-gp` switch.

Permission Functional Restrictions

The `-gd` server command line option determines the permission level required to start databases. By default, this option is set to `DBA`, so that only users with the SERVER OPERATOR system privilege can start SAP IQ databases. However, you can also set this option to `ALL` or `NONE`. `ALL` means that all users can start a database. `NONE` means that no users, including the user with the SERVER OPERATOR system privilege, can start a database.

3.7.2 Stopping Databases

Different ways to stop a database.

- Disconnect from a database started by a connection string. The database stops automatically when the last user disconnects from it, unless you explicitly set the `AUTOSTOP` connection parameter to NO.
- From Interactive SQL or Embedded SQL, use the `STOP DATABASE` statement.

See *STOP DATABASE Statement [Interactive SQL]* in *SAP IQ SQL Reference*.

3.8 Starting the iqdemo Database

Use the script provided at installation to create the `iqdemo` database and the configuration file to start it.

This configuration file, called `iqdemo.cfg`, contains all the parameters necessary to start the demo database. See *Getting Started With SAP IQ*.

4 Connect to SAP IQ Servers and Databases

SAP IQ runs in a client/server environment, in which many users can connect to a database server across a network.

You may be able to connect to more than one database server; specify connection options that clearly identify servers.

> **i** Note
>
> You can connect from Interactive SQL on a Windows or Linux client to SAP IQ on a UNIX server.

Client applications can connect to databases from ODBC, OLE DB, Embedded SQL applications, and Interactive SQL.

Any client application that uses a database must establish a connection to that database before any work can be done. The connection forms a channel through which all activity from the client application takes place. For example, your user ID determines permissions to carry out actions on the database—and the database server has your user ID because it is part of the request to establish a connection.

In this section:

Ways to Connect [page 39]
Interactive SQL and client applications use the same connection parameters.

Supported Connection Interfaces [page 50]
To establish a connection, the client application calls functions in one of the supported interfaces.

Connection Status [page 51]
Some client tools may not clearly indicate connection status. A failed command is your first indication that the connection does not exist.

How Connection Parameters Work [page 52]
When an application connects to a database, it uses a set of connection parameters to define the connection.

Connection Parameters in ODBC Data Sources [page 54]
Many client applications, including application development systems, use the ODBC interface to access SAP IQ.

Drivers for Connections [page 55]
When you work with a database, all your requests and commands go through a driver to the database itself.

File Data Sources [page 56]
On Windows operating systems, ODBC data sources are typically stored in the system registry. File data sources are an alternative, which are stored as files. File data sources are supported on both Windows and UNIX systems.

Connections from Other Databases [page 58]

You can access data in SAP IQ tables as a foreign data source from SAP Adaptive Server Enterprise. To take advantage of this feature, use the Component Integration Services (CIS) interface, which makes data from distributed, heterogeneous sources available to clients.

How to Test Connections [page 59]
The `dbping` command-line utility is provided to help troubleshoot connections. In particular, you can use it to test if a server with a certain name is available on your network.

Integrated Logins [page 60]
The integrated login feature allows you to maintain a single user ID and password for both database connections and operating system and/or network logins.

Connection Pooling [page 66]
Connection pooling may improve the performance of applications that make multiple, brief connections to the database server. If connection pooling is enabled for a connection, when it is disconnected, the connection is automatically cached and may be reused when the application reconnects. You control connection pooling with the ConnectionPool (CPOOL) connection parameter. Once an application makes a specified number of connections with the same connection string, then the connection is pooled.

Temporary Connections [page 68]
Temporary connections perform operations like running backups or initializing databases.

Logical Server Configuration [page 69]
In a multiplex, you can only access servers by using logical servers.

How to End Connections [page 71]
Connections must be ended under certain circumstances.

Connection Logging [page 72]
By default, each time a user connects or disconnects from a database, the `<dbname>.iqmsg` log records this action.

4.1 Ways to Connect

Interactive SQL and client applications use the same connection parameters.

Topic	Reference
Connect from client application	Supported Connection Interfaces [page 50]
Connect from Interactive SQL	Connecting to the Demo Database from dbisql (UNIX) [page 41] Connecting to the Demo Database from dbisql (Windows) [page 42]
Customize a connection string	Connection and Communication Parameters Reference [page 465]
Use data sources	Connecting Using a Data Source [page 48]

Topic	Reference
Diagnose network connection issues	Troubleshooting Network Communications [page 440]

In this section:

Connecting to the Demo Database from dbisql (UNIX) [page 41]
You do not need to specify the host and port for a database on your local machine.

Connecting to the Demo Database from dbisql (Windows) [page 42]
Many examples and exercises throughout the documentation start by connecting to the demo database from Interactive SQL, also called `dbisql`.

How Database Status Affects Local Connections [page 42]
The simplest connection scenario is when the database you want to connect to resides on your own machine.

Connecting to an Embedded Database [page 43]
An embedded database, designed for use by a single application, runs on the same machine as the application and is largely hidden from the application user.

TCP/IP protocol [page 44]
TCP/IP is used to connect clients to databases running on different computers.

Connecting Using a Data Source [page 48]
You can save sets of connection parameters in a data source. ODBC and JDBC using the iAnywhere JDBC driver use data sources, as do Embedded SQL applications like Interactive SQL Classic.

Connecting from SAP IQ Utilities [page 49]
SAP IQ database utilities that communicate with the server (rather than acting directly on database files) do so using Embedded SQL.

Default Connection Parameters [page 49]
You can leave many connection parameters unspecified, and instead use the default behavior to make a connection.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]

Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.1.1 Connecting to the Demo Database from dbisql (UNIX)

You do not need to specify the host and port for a database on your local machine.

Prerequisites

Make sure that your PATH and other environment variables are correctly set.

Procedure

1. At the UNIX prompt, to ensure that the demo database is loaded on a running server, enter:

   ```
   ps -eaf | grep iqdemo
   cd $IQDIR16/demo
   ```

2. To start the server the demo database:

   ```
   start_iq @iqdemo.cfg iqdemo.db
   ```

3. Start Interactive SQL:

   ```
   dbisql -c "uid=<userID>;pwd=<password>" -host <hostname> -port <portnum> -n
   <servername> <dbfilename>.db
   ```

   The `-c` parameter specifies connection parameters.

   For example, to connect to the demo database on remote host `host2` using a user name of `DBA` and password of `myp@ssword`, enter:

   ```
   dbisql -c "uid=DBA;pwd=myp@ssword" -host host2 -port 1870 -n host2_iqdemo
   $IQDIR16/demo/iqdemo.db
   ```

4.1.2 Connecting to the Demo Database from dbisql (Windows)

Many examples and exercises throughout the documentation start by connecting to the demo database from Interactive SQL, also called `dbisql`.

Context

Use the same process to connect to any database server that is already running. You can also specify a nondefault character set and language.

See the *SAP IQ Interactive SQL Guide*.

Procedure

1. Select **Programs** > **SAP** > **SAP IQ 16.0** > **Interactive SQL**.
2. On the Identification tab, type `DBA` and `sql` for the User and Password.
3. On the Database tab, choose **Find**.
4. Select your iqdemo server from the Find Servers screen and click **OK**.

4.1.3 How Database Status Affects Local Connections

The simplest connection scenario is when the database you want to connect to resides on your own machine.

If this is the case:

- Is the database already running on a server? If so, you can specify fewer parameters in the Connect dialog. If not, identify the database file so that Interactive SQL can start it.
- Are there multiple databases running on your machine? If so, identify the database to which you want Interactive SQL to connect. If there is only one database, Interactive SQL connects to it, and you do not need to specify it in the Connect dialog.

4.1.4 Connecting to an Embedded Database

An embedded database, designed for use by a single application, runs on the same machine as the application and is largely hidden from the application user.

Prerequisites

Make sure that your PATH and other environment variables are set.

Procedure

1. To start SAP IQ with no databases running, type **dbisql** at a system command prompt..
2. Type **CONNECT** in the command window, and press F9 to execute.
3. If you have an ODBC data source for your database, select that data source in the connection dialog.
4. Enter the user ID **DBA** and password **<password>**.
5. On the Database tab, enter the full path of the demo database in the **Database File** field. For example:

   ```
   /SAP/IQ-16_1/demo/iqdemo.db
   ```

6. Leave all other fields blank, and click **OK**.
   SAP IQ starts and loads the demo database, and Interactive SQL connects to the database.

In this section:

Connection Parameters to Customize Startup [page 43]
When an application uses an embedded database, the database is generally not running when the application connects.

4.1.4.1 Connection Parameters to Customize Startup

When an application uses an embedded database, the database is generally not running when the application connects.

You can start the database using the connection string, and by specifying the database file in the DatabaseFile (DBF) parameter of the connection string.

Database File Parameter

The DBF parameter specifies which database file to use. The database file automatically loads onto the default server, or starts a server if none is running.

The database unloads when there are no more connections to the database (generally when the application that started the connection disconnects). If the connection started the server, it stops once the database unloads.

Use these connection parameters to load the demo database as an embedded database, where <path> is the name of your SAP IQ installation directory:

```
dbf=path\iqdemo.db
uid=DBA
pwd=<password>
```

Start Parameter

The following connection parameters show how to customize the startup of the demo database as an embedded database. You may find this useful if you use command-line options, such as the cache size:

```
Start=start_iq -gd all
-gl all -gm 10 -gn 25 -gp 4096 -c 32M
-ti 4400 -tl 300
dbf=path\iqdemo.db
uid=DBA
pwd=<password>
```

4.1.5 TCP/IP protocol

TCP/IP is used to connect clients to databases running on different computers.

When you use the TCP/IP protocol, you can secure client/server communications using transport layer security and RSA encryption technology.

UDP

UDP is a transport layer protocol that sits on top of IP. UDP might be used to do initial server name resolution and then TCP is used for connection and communication.

UDP packets sent by the database server in response to client broadcasts contain no sensitive information. The data contained in these packets is limited to:

- the database server name
- the port number
- the database server version
- the names of databases running on the database server

You can hide database names from UDP broadcast responses by using the -dh database option.

You can specify the -sb server option to disable the UDP listeners.

Using TCP/IP with Windows

The TCP/IP implementation for database servers on Windows uses Winsock 2.2.

In this section:

Client/server communications encryption over TCP/IP [page 45]
You can secure communications between client applications and the database server over TCP/IP using Transport Layer Security (TLS).

IPv6 support [page 45]
On IPv6-enabled computers, the network database server listens by default on all IPv4 and IPv6 addresses.

Firewall connections [page 46]
There are restrictions on connections when the client application is on the other side of a firewall from the database server.

Dial-up network connections (CommLinks connection parameter) [page 47]
To connect over a dial-up network connection, you must use the CommLinks connection parameter.

4.1.5.1 Client/server communications encryption over TCP/IP

You can secure communications between client applications and the database server over TCP/IP using Transport Layer Security (TLS).

By default, communication packets are not encrypted, which poses a potential security risk. Transport Layer Security (TLS) provides server verification, strong encryption using RSA encryption technology, and other features for protecting data integrity.

4.1.5.2 IPv6 support

On IPv6-enabled computers, the network database server listens by default on all IPv4 and IPv6 addresses.

Usually no changes are required to the database server start line to use IPv6. In a case where specifying an IP address is required, the database server and the client libraries both accept IPv4 and IPv6 addresses. For example, if a computer has more than one network card enabled, it probably has two IPv4 addresses and two IPv6 addresses. For IPv6 addresses that include a port number, you must enclose the address in either square brackets or parentheses. If you want the database server to listen on only one of the IPv6 addresses, you can specify an address in the following format:

```
start_iq -x tcpip(MyIP=fd77:55f:5a64:52a:202:5445:5245:444f) ...
```

Similarly, if a client application needs to specify the IP address of a server, the connection string or ODBC data source can contain the address, in the following format:

```
...HOST=fd77:55f:5a64::444f;...
```

Each interface is given an interface identifier, which appears at the end of an IPv6 address. For example, if `ipconfig.exe` lists the address `fd77:55f:5a64::444f%7`, the interface identifier is 7. When specifying an IPv6 address on a Windows platform, the interface identifier should be used. On Unix, you can specify either an interface identifier or an interface name (the interface name is the name of the interface reported by ifconfig). For example, the interface name is eth1 in the following IPv6 address: `fd77:55f:5a64::444f%eth1`. An interface identifier is required when specifying IPv6 addresses on Linux (kernel 2.6.13 and later). This requirement affects values specified by the following:

- Host connection parameter
- Broadcast (BCAST) protocol option
- Host (IP) protocol option (client side only)
- MyIP (ME) protocol option

Example

Suppose `ipconfig.exe` lists two interfaces, one with the identifier 1 and the other with the identifier 2. If you are looking for a database server that is on the network used by interface number 2, you can tell the client library to broadcast only on that interface:

```
LINKS=tcpip(BROADCAST=ff02::1%2)
```

The IPv6 link-local multicast address is `ff02::1`.

4.1.5.3 Firewall connections

There are restrictions on connections when the client application is on the other side of a firewall from the database server.

Firewall software may filter network packets according to network port. Also, it is common to disallow UDP packets from crossing the firewall.

Usually, you can connect through a properly configured firewall using the Host connection parameter and providing the database server address and port. If the database server is using the default port of 2638, the port is not required.

The following connection string fragment connects to a database server named myserver running on a computer at address serverhost using port 2020. No UDP packets are used because the Host connection parameter specifies the TCP/IP address and port.

```
Server=myserver;Host=serverhost:2020
```

Firewalls that only allow certain client ports

If the firewall must be configured to allow only certain client ports, then you must use the CommLinks(LINKS) connection parameter instead of the Host connection parameter. The following TCP/IP protocol options are required when you use the CommLinks connection parameter:

Host

Set this protocol option to the host name on which the database server is running. You can use the short form IP.

ServerPort

If your database server is not using the default port of 2638, you must specify the port it is using. You can use the short form PORT.

ClientPort

Set this protocol option to a range of allowed values for the client application to use. You can use the short form CPORT.

DoBroadcast=NONE

Set this protocol option to prevent UDP from being used when connecting to the server. You can use the short from DOBROAD.

Your firewall must be configured to allow TCP/IP traffic between the database server's address and all the clients' addresses. The database server's address is the IP address of the computer running the server (the HOST protocol option) and the database server's IP port number (the ServerPort protocol option, default 2638).

Use a range with more client ports than the maximum number of concurrent connections from each client computer since there is a several minute timeout before a client port can be reused. The range of clients specified in the ClientPort protocol option must match the range allowed by the firewall.

Example

The following connection string fragment restricts the client application to ports 5050 through 5060, and connects to a database server named myeng running on the computer at address myhost using the server port 2020. No UDP broadcast is performed because the DoBroadcast protocol option is set to NONE.

```
Server=myeng;LINKS=tcpip(ClientPort=5050-5060;HOST=myhost;PORT=2020;DoBroadcast=N
ONE)
```

4.1.5.4 Dial-up network connections (CommLinks connection parameter)

To connect over a dial-up network connection, you must use the CommLinks connection parameter.

On the client side, you should specify the following protocol options:

Host

You should specify the host name or IP address of the database server using the Host (IP) protocol option.

DoBroadcast

If you specify the Host (IP) protocol option, there is no need to do a broadcast search for the database server. For this reason, use direct broadcasting.

MyIP

You should set **`MyIP=NONE`** on the client side.

TIMEOUT

Set the Timeout (TO) protocol option to increase the time the client waits while searching for a server.

Example

The following is an example of a typical CommLinks (LINKS) connection parameter used to connect over a dialup network connection:

```
LINKS=tcpip(MyIP=NONE;DoBroadcast=DIRECT;HOST=<server-ip>); TIMEOUT=15)
```

4.1.6 Connecting Using a Data Source

You can save sets of connection parameters in a data source. ODBC and JDBC using the iAnywhere JDBC driver use data sources, as do Embedded SQL applications like Interactive SQL Classic.

Context

The iqdemo data source holds a set of connection parameters, including the database file and a start parameter to start the demo database. The server name in this data source is "<hostname>_iqdemo" where hostname represents your system name.

Create data sources from the ODBC Administrator. See *Creating an ODBC Data Source* in the *SAP IQ Installation and Update Guide*.

See *Connecting Using an ODBC Data Source* in *SAP IQ Interactive SQL Guide*.

4.1.7 Connecting from SAP IQ Utilities

SAP IQ database utilities that communicate with the server (rather than acting directly on database files) do so using Embedded SQL.

How Database Utilities Obtain Connection Parameter Values

Many of the administration utilities obtain the connection parameter values by:

Using values specified on the command line (if any). For example, the following command takes a backup of the catalog store on the `demo` database, using the user ID `DBA` and the password `sql`:

```
dbbackup -y -x -c
'uid=DBA;pwd=<password>;eng=iqdemo;dbn=iqdemo.db;links=tcpip{port=2638}' -d  '/
mydir'
```

1. Using the SQLCONNECT environment variable settings if any command line values are missing. SAP IQ database utilities do not set this variable automatically. This option provides better password security than other methods. For a description of the SQLCONNECT environment variable, see *SAP IQ SQL Reference* > *File Locations and Installation Settings* > *Environment Variables*.
2. Prompting you for a user ID and password to connect to the default database on the default server, if parameters are not set in the command line or the SQLCONNECT environment variable.

For a description of command line options for each database utility, see the *SAP IQ Utility Reference*.

4.1.8 Default Connection Parameters

You can leave many connection parameters unspecified, and instead use the default behavior to make a connection.

> ⚠ Caution
>
> Avoid relying on default behavior in production environments, especially if you distribute your application to customers who may install other SAP IQ or SAP SQL Anywhere applications on their machine.

Default Database Server

If you are connecting to a database on your local server, and more than one database has been started on that server, you need to specify the database you wish to connect to, but you can leave the server as a default:

```
dbn=<db_name>
uid=<user_id>
pwd=<password>
```

> ⚠ Caution
>
> Do not use these parameters if more than one local server is running, or you may connect to the wrong server.

Default Database

If more than one server is running, specify which one you wish to connect to. If only one database has been started on that server, you don't need to specify the database name. The following connection string connects to a named server, using the default database:

```
eng=<server_name>
uid=<user_id>
pwd=<password>
```

Connections Without Defaults

The following connection string connects to a named server, using a named database:

```
eng=<server_name>
dbn=<db_name>
uid=<user_id>
pwd=<password>
```

4.2 Supported Connection Interfaces

To establish a connection, the client application calls functions in one of the supported interfaces.

SAP IQ supports the following interfaces:

- ODBC
- OLE DB
- Embedded SQL
- SAP Open Client – see *SAP IQ as a Data Server for Client Applications* in *SAP IQ Programming Reference*.
- JDBC – see *JDBC CLI* in *SAP IQ Programming Reference*. Interactive SQL supports the iAnywhere JDBC driver, which is included with SAP IQ.

  > ℹ Note
  >
  > The iAnywhere JDBC driver is deprecated.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.3 Connection Status

Some client tools may not clearly indicate connection status. A failed command is your first indication that the connection does not exist.

Context

A quick way to confirm the connection is by querying the database name using one of the following methods.

Procedure

- To display the current database:

```
select db_name()
```

- To specify a different database:

```
select db_name([ <database_id> ])
```

Task overview: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.4 How Connection Parameters Work

When an application connects to a database, it uses a set of connection parameters to define the connection.

Connection parameters include information such as the server name, the database name, and a user ID.

A keyword-value pair, of the form `<parameter=value>`, specifies each connection parameter. For example, you specify the password connection parameter for the default password as follows:

```
Password=<password>
```

Connection parameters are assembled into connection strings. In a connection string, a semicolon separates each connection parameter, as follows:

```
ServerName=host_iqdemo;DatabaseName=iqdemo
```

Several connection parameters affect how a server is started. It is recommended that you use the following connection parameters instead of providing the corresponding server options within the StartLine (START) connection parameter:

- EngineName (ENG)
- DatabaseFile (DBF)

In this section:

Format for Connection Strings [page 53]
Examples in this document represent connection strings in a specific form.

How Applications Pass Connection Parameters [page 53]
Connection parameters are passed to the interface library as a connection string.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.4.1 Format for Connection Strings

Examples in this document represent connection strings in a specific form.

The connection strings are represented in the following ways:

```
<parameter1>=<value1>
<parameter2>=<value2>
...
```

The example is equivalent to the following connection string:

```
<parameter1>=<value1>;<parameter2>=<value2>
```

You must enter a connection string on a single line, with the parameter settings separated by semicolons.

4.4.2 How Applications Pass Connection Parameters

Connection parameters are passed to the interface library as a connection string.

This string consists of a set of parameters, separated by semicolons.

In general, the connection string built up by an application and passed to the interface library does not correspond directly to the way a user enters the information. Instead, a user may fill in a dialog box, or the application may read connection information from an initialization file.

Certain SAP IQ utilities accept a connection string as the `-c` command-line option and pass the connection string on to the interface library without change. For example, to stop a database named `iqdemo` on the server `myserver`, enter:

```
dbstop -c "uid=DBA;pwd=<password>;eng=myserver;dbn=iqdemo"
```

> **Note**
>
> Interactive SQL processes the connection string internally. It does not simply pass on the connection parameters to the interface library. Do not use Interactive SQL to test connection strings from a command prompt.

4.5 Connection Parameters in ODBC Data Sources

Many client applications, including application development systems, use the ODBC interface to access SAP IQ.

When connecting to the database, ODBC applications typically use ODBC data sources. An ODBC data source is a set of connection parameters, stored in the registry or in a file.

For SAP IQ, ODBC data sources can be used not only by ODBC applications on Windows, but also by other applications:

- SAP IQ client applications on UNIX can use ODBC data sources, as well as those on Windows operating systems. On UNIX, the data source is stored as a file.
- SAP IQ client applications using the OLE DB or Embedded SQL interfaces can use ODBC data sources, as well as ODBC applications.
- Interactive SQL can use ODBC data sources.
- JDBC connections using the iAnywhere JDBC driver can use ODBC data sources.

> **Note**
>
> The iAnywhere JDBC driver is deprecated.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]

How Connection Parameters Work [page 52]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.6 Drivers for Connections

When you work with a database, all your requests and commands go through a driver to the database itself.

SAP IQ Programming Reference describes supported drivers and programming interfaces.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.7 File Data Sources

On Windows operating systems, ODBC data sources are typically stored in the system registry. File data sources are an alternative, which are stored as files. File data sources are supported on both Windows and UNIX systems.

In Windows, file data sources typically have the extension `.dsn`. They consist of sections, each section starting with a name enclosed in square brackets. DSN files are very similar in layout to initialization files.

To connect using a File Data Source, use the `FileDSN` connection parameter. You cannot use both `DSN` and `FileDSN` in the same connection.

File Data Sources Can Be Distributed

One benefit of file data sources is that you can distribute the file to users, so that connection information does not have to be reconstructed on each machine. If the file is placed in the default location for file data sources, it is picked up automatically by ODBC. In this way, managing connections for many users can be made simpler.

> **i** Note
>
> Because DSNs are stored in the Windows registry, they are public information. For this reason, you should not put a password in a DSN, unless you encrypt it. If you want to store your password in your data source, use a File DSN.

Embedded SQL applications can also use ODBC file data sources.

In this section:

Creating a File Data Source Using the ODBC Administrator [page 57]
You can use the ODBC Administrator tool to create a File Data Source.

File Data Sources and Text Editors [page 57]
A file data source is a text file that can be edited with any text editor. One limitation to using a text editor is that you cannot store encrypted passwords in the file.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]

Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.7.1 Creating a File Data Source Using the ODBC Administrator

You can use the ODBC Administrator tool to create a File Data Source.

Procedure

1. Start the ODBC Administrator, click the **File DSN** tab and click **Add**.
2. Select SAP IQ from the list of drivers, and click **Next**.
3. Follow the instructions to create the data source.

4.7.2 File Data Sources and Text Editors

A file data source is a text file that can be edited with any text editor. One limitation to using a text editor is that you cannot store encrypted passwords in the file.

Example of a File Data Source

```
[Sample File Data Source]
 ENG = iqdemo
 DBA = DBA
 PWD = MyIQD@tabase
```

See *SAP IQ Utility Reference > iqdsn Database Administration Utility*.

4.8 Connections from Other Databases

You can access data in SAP IQ tables as a foreign data source from SAP Adaptive Server Enterprise. To take advantage of this feature, use the Component Integration Services (CIS) interface, which makes data from distributed, heterogeneous sources available to clients.

With CIS, define “proxy tables” in SAP ASE that represent your SAP IQ tables. You can then query the proxy tables from SAP ASE. See *Component Integration Services Users Guide for SAP Adaptive Server Enterprise*.

CIS and SAP IQ offer several other ways to connect to other databases and share data, so that user applications can access your entire data warehouse through a common interface. Using CIS, you can:

- Access data in an SAP ASE database from SAP IQ. This functionality is supported on only some platforms. See the *SAP IQ Installation and Update Guide* for your platform.
- Access data in SAP IQ and SAP SQL Anywhere databases on other database servers.
- Access other foreign data sources, including other vendors' relational databases, Excel spreadsheet data, and text files.
- Join tables from separate SAP IQ databases.

In this section:

Avoiding Port Number Conflicts on UNIX [page 59]
Update configuration files to avoid port number conflicts.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
How to Test Connections [page 59]
Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]
Access Remote Data [page 303]

4.8.1 Avoiding Port Number Conflicts on UNIX

Update configuration files to avoid port number conflicts.

Procedure

1. Add the following line to `$IQDIR16/scripts/default.cfg` with an unused port number, for example:

   ```
   -x  tcpip{port=4444}
   ```

2. Look for a port number definition in each configuration file. For example, `/usr/summers/mydemo/iqdemo.cfg` contains the following line:

   ```
   -x  tcpip{port=2638}
   ```

3. Edit the line and replace the default port number with the new one, for example:

   ```
   -x  tcpip{port=4444}
   ```

4. Save each file when finished.

Results

If SAP SQL Anywhere is on the same subnet as SAP IQ, the server names must be unique.

4.9 How to Test Connections

The `dbping` command-line utility is provided to help troubleshoot connections. In particular, you can use it to test if a server with a certain name is available on your network.

The `dbping` utility takes a connection string as a command line option, but by default, only those pieces required to locate a server are used. It does not attempt to start a server.

The following line tests to see if a server named Ciaran is available over a TCP/IP connection:

```
dbping -c "eng=Ciaran;CommLinks=tcpip"
```

The following command tests to see if a default server is available on the current machine:

```
dbping
```

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.10 Integrated Logins

The integrated login feature allows you to maintain a single user ID and password for both database connections and operating system and/or network logins.

Operating Systems Supported

Integrated login capabilities are available for the Windows server only. It is possible for clients on supported Windows platforms to use integrated logins to connect to a network server running on Windows.

Benefits of Integrated Logins

An integrated login is a mapping from one or more Windows user profiles to an existing user in a database. A user who has successfully navigated the security for that user profile and logged in to their machine can connect to a database without providing an additional user ID or password.

To accomplish this, the database must be enabled to use integrated logins and a mapping must have been granted between the user profile used to log in to the machine and/or network, and a database user.

Using an integrated login is more convenient for the user and permits a single security system for database and network security. Its advantages include:

- When connecting to a database using an integrated login, the user does not need to enter a user ID or password.

- If you use an integrated login, the user authentication is done by the operating system, not the database: a single system is used for database security and machine or network security.
- Multiple user profiles can be mapped to a single database user ID.
- The name and password used to log in to the Windows machine do not have to match the database user ID and password.

> ⚠ Caution
>
> Integrated logins offer the convenience of a single security system but there are important security implications which database administrators should be familiar with.

In this section:

Using Integrated Logins [page 62]
You perform several steps to connect via an integrated login.

Security Concerns: Unrestricted Database Access [page 64]
The integrated login feature uses the login control system of Windows instead of the system that SAP IQ uses to control access to the database.

Temporary Public Options Provide Added Security [page 65]
Setting the value of the LOGIN_MODE option for a given database to `Mixed` (deprecated) or `Integrated` using the following SQL statement permanently enables integrated logins for that database.

Network Aspects of Integrated Logins [page 66]
If the database is located on a network server, then one of two conditions must be met for integrated logins to be used.

Default Integrated Login Users [page 66]
You can create a default integrated login user ID that allows connection via an integrated login to succeed even if no integrated login mapping exists for the user profile currently in use.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Connection Pooling [page 66]
Temporary Connections [page 68]

Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.10.1 Using Integrated Logins

You perform several steps to connect via an integrated login.

Context

Details of these steps are described in the following sections.

Procedure

1. Enable the integrated login feature in a database by setting the value of the LOGIN_MODE database option to either `Mixed` (deprecated) or `Integrated` (the option is case-insensitive), in place of the default value of `Standard`. This step requires SET ANY SECURITY OPTION system privilege.
2. Create an integrated login mapping between a user profile and an existing database user. You can do this using a SQL statement.
3. Connect from a client application in such a way that the integrated login facility is triggered.

In this section:

Enabling the Integrated Login Feature [page 63]
The LOGIN_MODE database option determines whether the integrated login feature is enabled. As database options apply only to the database in which they are found, different databases can have a different integrated login setting even if they are loaded and running within the same server.

Creating an Integrated Login in SQL [page 63]
User profiles can only be mapped to an existing database user ID. When that database user ID is removed from the database, all integrated login mappings based on that database user ID are automatically removed.

Revoking an Integrated Login Permission (SQL) [page 64]
You can remove an integrated login mapping using Interactive SQL.

Connecting from a Client Application [page 64]
A client application can connect to a database using an integrated login in one of the following ways:

4.10.1.1 Enabling the Integrated Login Feature

The LOGIN_MODE database option determines whether the integrated login feature is enabled. As database options apply only to the database in which they are found, different databases can have a different integrated login setting even if they are loaded and running within the same server.

Procedure

Choose one of the following three (case-sensitive) values for the LOGIN_MODE database option:

Option	Description
Standard	The default setting, which does not permit integrated logins. An error occurs if an integrated login connection is attempted.
Mixed	Both integrated logins and standard logins are allowed.
Integrated	With this setting, all logins to the database must be made using integrated logins. ⚠ Caution Setting the LOGIN_MODE database option to Integrated restricts connections to only those users who have been granted an integrated login mapping. Attempting to connect using a user ID and password generates an error. The only exception to this are users with MANAGE ANY USER system privilege.

4.10.1.2 Creating an Integrated Login in SQL

User profiles can only be mapped to an existing database user ID. When that database user ID is removed from the database, all integrated login mappings based on that database user ID are automatically removed.

Procedure

The following SQL statement allows Windows users fran_whitney and matthew_cobb to log in to the database as the user `DBA`, without having to know or provide the DBA user ID or password:

```
GRANT INTEGRATED LOGIN
TO fran_whitney, matthew_cobb
AS USER DBA
```

For more information, see *SAP IQ SQL Reference* > *SQL Statements* > *GRANT INTEGRATED LOGIN Statement*.

4.10.1.3 Revoking an Integrated Login Permission (SQL)

You can remove an integrated login mapping using Interactive SQL.

Procedure

1. Connect to a database as a user with administrative authority over the MANAGE ANY USER system privilege.
2. Execute a REVOKE INTEGRATED LOGIN FROM statement.

4.10.1.4 Connecting from a Client Application

A client application can connect to a database using an integrated login in one of the following ways:

Context

- Set the INTEGRATED parameter in the list of connection parameters to `yes`.
- Specify neither a user ID nor a password in the connection string or connection dialog. This method is available only for Embedded SQL applications, including the SAP IQ administration utilities.

If `INTEGRATED=yes` is specified in the connection string, an integrated login is attempted. If the connection attempt fails and the LOGIN_MODE database option is set to `Mixed` (deprecated), the server attempts a standard login.

If an attempt to connect to a database is made without providing a user ID or password, an integrated login is attempted. The attempt succeeds or fails depending on whether the current user profile name matches an integrated login mapping in the database.

4.10.2 Security Concerns: Unrestricted Database Access

The integrated login feature uses the login control system of Windows instead of the system that SAP IQ uses to control access to the database.

Essentially, you pass through the database security if you can log in to the machine hosting the database, and if other conditions outlined in this chapter are met.

If you successfully log in to the Windows server as "dsmith", you can connect to the database without any further proof of identification provided there is either an integrated login mapping or a default integrated login user ID.

When using integrated logins, database administrators should give special consideration to the way Windows enforces login security in order to prevent unwanted access to the database.

In particular, be aware that by default a "Guest" user profile is created and enabled when Windows Workstation or Server is installed.

> ⚠ Caution
>
> Leaving the user profile Guest enabled can permit unrestricted access to a database being hosted by that server.

If the Guest user profile is enabled and has a blank password, any attempt to log in to the server will be successful. It is not required that a user profile exist on the server, or that the login ID provided have domain login permissions. Literally any user can log in to the server using any login ID and any password: they are logged in by default to the Guest user profile.

This has important implications for connecting to a database with the integrated login feature enabled.

Consider the following scenario, which assumes the Windows server hosting a database has a "Guest" user profile that is enabled with a blank password.

- An integrated login mapping exists between the user `dsmith` and the database user ID `DBA`. When the user `dsmith` connects to the server with her correct login ID and password, she connects to the database as `DBA`, a user with full administrative rights.
- But anyone else attempting to connect to the server as "dsmith" will successfully log in to the server regardless of the password they provide because Windows will default that connection attempt to the "Guest" user profile. Having successfully logged in to the server using the "dsmith" login ID, the unauthorized user successfully connects to the database as `DBA` using the integrated login mapping.

> ℹ Note
>
> Disable the "Guest" user profile for security. The safest integrated login policy is to disable "Guest" on any Windows machine hosting a SAP IQ database. This can be done using the Windows User Manager utility.

4.10.3 Temporary Public Options Provide Added Security

Setting the value of the LOGIN_MODE option for a given database to `Mixed` (deprecated) or `Integrated` using the following SQL statement permanently enables integrated logins for that database.

```
SET OPTION Public.LOGIN_MODE = Mixed
```

If the database is shut down and restarted, the option value remains the same and integrated logins are still enabled.

Changing the LOGIN_MODE option temporarily will still allow user access via integrated logins. The following statement will change the option value temporarily:

```
SET TEMPORARY OPTION "Public".LOGIN_MODE = Mixed
```

If the permanent option value is `Standard`, the database will revert to that value when it is shut down.

Setting temporary public options can be considered an additional security measure for database access since enabling integrated logins means that the database is relying on the security of the operating system on which it is running. If the database is shut down and copied to another machine (such as a user's machine) access to

the database reverts to the SAP IQ security model and not the security model of the operating system of the machine where the database has been copied.

See *SAP IQ SQL Reference > SQL Statements > SET OPTION Statement*.

4.10.4 Network Aspects of Integrated Logins

If the database is located on a network server, then one of two conditions must be met for integrated logins to be used.

Conditions are:

- The user profile used for the integrated login connection attempt must exist on both the local machine and the server. As well as having identical user profile names on both machines, the passwords for both user profiles must also be identical.
  For example, when the user `jsmith` attempts to connect using an integrated login to a database loaded on network server, identical user profile names and passwords must exist on both the local machine and application server hosting the database. `jsmith` must be permitted to log in to both the local machine and the server hosting the network server.
- If network access is controlled by a Microsoft Domain, the user attempting an integrated login must have domain permissions with the Domain Controller server and be logged in to the network. A user profile on the network server matching the user profile on the local machine is not required.

4.10.5 Default Integrated Login Users

You can create a default integrated login user ID that allows connection via an integrated login to succeed even if no integrated login mapping exists for the user profile currently in use.

For example, if no integrated login mapping exists for the user profile name `jsmith`, an integrated login connection attempt will normally fail when `jsmith` is the user profile in use.

However, if you create a user ID named `Guest` in a database, an integrated login successfully maps to the `Guest` user ID if no integrated login mapping explicitly identifies the user profile `jsmith`.

The default integrated login user permits anyone attempting an integrated login to successfully connect to a database if the database contains a user ID named `Guest`. The permissions and privileges granted to the newly-connected user are determined by the privileges granted to the `Guest` user ID.

4.11 Connection Pooling

Connection pooling may improve the performance of applications that make multiple, brief connections to the database server. If connection pooling is enabled for a connection, when it is disconnected, the connection is automatically cached and may be reused when the application reconnects. You control connection pooling with the ConnectionPool (CPOOL) connection parameter. Once an application makes a specified number of connections with the same connection string, then the connection is pooled.

An application must make five connections with the same connection string before a connection is cached. The connection name can be different each time, but all other connection parameters must be identical for a cached connection to be reused.

If the application process connects again and there are cached connections available for the same connection string, the cached connection is reused. Connections remain in the cached state for the time specified by the ConnectionPool (CPOOL) connection parameter (60 seconds by default).

Cached connections are not reused if it would change the behavior of the application. For example, cached connections are not reused:

- For databases that stop automatically when there are no connections to them.
- If connections are disabled.
- If the database server has reached its connection limit.
- If a password has changed.
- If the login_mode option is set.

To ensure that connection pooling is transparent to the application, a connection is disconnected if a failure occurs when caching a connection. If a failure occurs when attempting to reuse a cached connection, the database server attempts to connect normally.

A connection is cached if it is disconnected and the maximum number of connections specified by the CPOOL connection parameter has not been reached. The connection is reinitialized, and the cached connection remains connected to the database server even though the application has disconnected it. The cleanup and reinitialization of a connection includes the following activities:

- Rolling back all outstanding transactions.
- Dropping temporary tables, temporary functions, and variables.
- Resetting connection options and connection counters.
- Decrementing and incrementing the database server connection counts. You are not informed that there are active connections when a database server with only cached connections shuts down.
- Executing all defined disconnect and connect events.
- Executing the login_procedure database option and verifying the login policy.
- Resetting the connection ID.

Using Connection Pooling with other Connection Pooling Products

If you are using a product or API that supports connection pooling, then the connection pooling of the product or API supersedes SAP IQ connection pooling. Both types of connection pooling can be active at the same time.

The behavior of connection pooling in your product or the API may be significantly different than SAP IQ connection pooling. If the behavior of connection pooling in your product or API is inappropriate for an application, SAP IQ connection pooling can be used and may improve the performance of some applications.

Connection Pooling and Read-Only Scale-Out

If the NodeType (NODE) connection parameter is also specified for a connection, the application typically connects to the primary server and the primary server determines which copy node is least heavily loaded. The

connection is then redirected to that node. If the application makes and drops several such connections within a short period of time, the connection is pooled and the primary server is not asked which copy node to use. This behavior reduces the load on the primary server, but may not give expected behavior.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.12 Temporary Connections

Temporary connections perform operations like running backups or initializing databases.

Use the system procedures `sa_conn_info` or `sa_conn_list` for information about temporary connections. The ParentConnection property returns the connection ID of the connection that spawned the temporary connection.

Temporary connections have connection IDs that are larger than 1 billion (1000000000), and their names describe the function of the connection.

The following example uses the `sa_conn_info` system procedure to return a result set showing which connection created a temporary connection.

```
SELECT Number, Name, ParentConnection FROM sa_conn_info();
```

Connection 8 spawned the temporary connection that executed a `CREATE DATABASE` statement.

```
Number        Name               ParentConnection
--------------------------------------------------------------
1000000048    INT: CreateDB      8
9             SQL_DBC_14675af8   (NULL)
```

```
8               SQL_DBA_152d5ac0    (NULL)
```

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]
Connection Pooling [page 66]
Logical Server Configuration [page 69]
How to End Connections [page 71]
Connection Logging [page 72]

4.13 Logical Server Configuration

In a multiplex, you can only access servers by using logical servers.

Not all member nodes may be available at all times to its logical server for reasons such as a failure or exclusion of a member node from the multiplex. At any given time, the effective logical server configuration represents the current dynamic constitution of the logical server consisting of all member nodes that are actually available for use to the logical server. The effective logical server configuration is essentially a function of static logical server configuration and dynamic state of the multiplex.

> **i** Note
>
> Only multiplex configurations support logical servers. Information about built-in logical servers and logical server policies can remain in the catalog in a simplex environment, but is not used.

In this section:

Connections in Simplex [page 70]
In simplex, connections are unaffected by the login policy setting of logical server assignments.

Connections in Multiplex [page 70]

In a multiplex, login policies control access to logical servers. All users of a login policy can only access those multiplex servers that are effective members of the assigned logical servers.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
How to End Connections [page 71]
Connection Logging [page 72]

4.13.1 Connections in Simplex

In simplex, connections are unaffected by the login policy setting of logical server assignments.

Connections have no logical server context. The base setting of login policy option `locked` is applied before a server accepts connections.

4.13.2 Connections in Multiplex

In a multiplex, login policies control access to logical servers. All users of a login policy can only access those multiplex servers that are effective members of the assigned logical servers.

When you connect to a multiplex server, logical server context of the connection is determined based on the effective logical server assignment of the user's login policy and the current node.

In this section:

Logical Server Context of a Connection [page 71]
When you establish a user connection, the user's login policy and the current node determine the logical server context of the connection.

4.13.2.1 Logical Server Context of a Connection

When you establish a user connection, the user's login policy and the current node determine the logical server context of the connection.

- When effective logical server assignment for the user's login policy is one or more logical servers, then the logical server context of the connection is determined based upon current node's unambiguous membership in one of the specified logical servers.

  > **i** Note
  >
  > Logical servers to which a given login policy allows access do not have overlapping membership.

- A connection fails if the current node is not a member of any of the logical servers assigned to the user's login policy.
- A connection also fails if the effective logical server assignment for the user's login policy is set to NONE.

4.14 How to End Connections

Connections must be ended under certain circumstances.

Connections end when:

- In Interactive SQL or Embedded SQL, a user or application issues an explicit DISCONNECT statement for the current connection, a specified connection, or all connections for that application.
- In Interactive SQL, select **SQL** > **Disconnect**.
- An application with active connections, such as Interactive SQL, is closed.

Users with DROP CONNECTION system privilege can also drop a specific connection in Interactive SQL or Embedded SQL by issuing a DROP CONNECTION statement.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]

Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
Connection Logging [page 72]

4.15 Connection Logging

By default, each time a user connects or disconnects from a database, the `<dbname>.iqmsg` log records this action.

You can control logging of user connections and disconnections using the database option, LOG_CONNECT. If connection logging is disabled when a user connects, and then turned ON before the user disconnects, the message log shows that user disconnecting but not connecting.

For more information, see *SAP IQ SQL Reference* > *Database Options* > *Alphabetical List of Options* > *LOG_CONNECT Option*.

Parent topic: Connect to SAP IQ Servers and Databases [page 38]

Related Information

Ways to Connect [page 39]
Supported Connection Interfaces [page 50]
Connection Status [page 51]
How Connection Parameters Work [page 52]
Connection Parameters in ODBC Data Sources [page 54]
Drivers for Connections [page 55]
File Data Sources [page 56]
Connections from Other Databases [page 58]
How to Test Connections [page 59]
Integrated Logins [page 60]
Connection Pooling [page 66]
Temporary Connections [page 68]
Logical Server Configuration [page 69]
How to End Connections [page 71]

5 Create and Manage SAP IQ Databases

Use Interactive SQL or SAP IQ Cockpit to create and manage SAP IQ databases.

In this section:

SAP IQ Database Design [page 73]
The right database design enhances data usability and the speed of data retrieval.

Create SAP IQ Databases [page 74]
To define your database, use SQL statements or a database design tool.

Dropping a Database [page 86]
Dropping a database deletes all tables and data from disk, including the transaction log that records alterations to the database. It also drops all of the dbspaces associated with the database.

Schema and Data Export [page 86]
The `iqunload` utility lets you make a full or partial copy of an existing database schema and then load it into another existing database of the same installation version. Data from the original database can be optionally included.

Disconnecting Other Users from a Database [page 92]
SAP IQ gives you the ability to disconnect other users from a given database.

Disconnecting from a Database in Embedded SQL [page 92]
When you are finished working with a database, you can disconnect a named connection or all connections.

Disconnecting All Connections from a Database in Interactive SQL [page 93]
When you are finished working with a database, you can disconnect a named connection or all connections.

Showing System Objects in Interactive SQL [page 93]
In Interactive SQL, you cannot query system tables, but you can browse the contents of a system view.

Setting Database Options in Interactive SQL [page 94]
Database options are configurable settings that change the way the database behaves or performs.

5.1 SAP IQ Database Design

The right database design enhances data usability and the speed of data retrieval.

Before you create the database, decide where to store the data, how much space your database requires, and who will be able to define or modify database objects. SAP IQ provides tools and processes to help.

SAP PowerDesigner can help you design your database, by building a conceptual, physical, or object-oriented data model, and then generating the database from the model. It also lets you reverse engineer, creating a model from an existing database.

No matter which design tool is used, the database administrator (DBA) generally designs the database and defines its contents. To create an effective design, the DBA needs to work with individuals throughout your organization to understand how data will be used. The DBA also needs to understand the concepts underlying SAP IQ databases.

An SAP IQ database is a relational database that is optimized for use as a data warehouse. As a relational database, it consists of a set of related tables that organize the data; as a data warehouse, it provides efficient access to very large sets of data by means of indexes.

When you create a database, you specify the structure of these tables, the types of data allowed in them, the relationships among tables, the indexes that store the table data, and views that control who has access to the data. Before creating a database, review the concepts in *Introduction to SAP IQ*.

5.2 Create SAP IQ Databases

To define your database, use SQL statements or a database design tool.

Some application design systems, such as SAP PowerDesigner, contain facilities for creating database objects. These tools construct SQL statements that are submitted to the server, typically through its ODBC interface. If you are using one of these tools, you do not need to construct SQL statements to create tables, assign permissions, and so on.

Database design tools such as SAP PowerDesigner provide a thorough and reliable approach to developing well-designed databases.

Database Creation with SAP IQ Cockpit

To create an SAP IQ database in SAP IQ Cockpit, see the SAP IQ Cockpit for SAP IQ online help in SAP IQ Cockpit or at https://help.sap.com/viewer/p/SAP_IQ.

In this section:

Database Creation with SQL [page 75]
You can use SQL statements for defining database objects directly if you build your database from an interactive SQL tool, such as `dbisql`.

IQ Main Store and IQ Temporary Store Space Management [page 85]
Options `MAIN_RESERVED_DBSPACE_MB` and `TEMP_RESERVED_DBSPACE_MB` provide room for checkpoint, commit, and release savepoint operations.

Related Information

Setting Up Windows Access to Raw Devices [page 104]

5.2.1 Database Creation with SQL

You can use SQL statements for defining database objects directly if you build your database from an interactive SQL tool, such as `dbisql`.

Even if you use an application design tool, you may want to use SQL statements to add features to the database if they are not supported by the design tool.

Once the database is created, you can connect to it and build the tables and other objects that you need in the database.

In this section:

Before You Create a Database [page 76]
Perform prerequisite actions before creating a database using SQL statements.

Starting the Utility Database [page 78]
You must start the utility database before creating new databases if no databases are built yet.

Dbspaces Created Automatically [page 78]
When you create a database, the database server creates certain dbspaces automatically.

Database File Placement [page 79]
When you create a database, consider whether you will ever need to move the database. The location of files for dbspaces may also affect performance.

CREATE DATABASE Statement Defaults [page 80]
The `CREATE DATABASE` statement has four required parameters and several optional parameters.

Increase Password Security [page 80]
Passwords are an important part of any database security system. There are several options for increasing password security.

Relative Path Names [page 81]
You can create a database using a relative or fully qualified path name for each of the files for the database.

IQ PATH Parameter Guidelines [page 81]
The required IQ PATH parameter tells SAP IQ that you are creating an SAP IQ database, not an SAP SQL Anywhere database.

IQ PAGE SIZE Parameter Guidelines [page 83]
The IQ PAGE SIZE parameter determines memory and disk use.

Database Size Guidelines [page 84]
When you create a database, you set the size and reserve size in MB of the initial database file (the `IQ_SYSTEM_MAIN` dbspace).

PAGE SIZE Parameter Guidelines [page 85]
Set the catalog store page size with the `CREATE DATABASE PAGE SIZE` option. The default and minimum value for this option is 4096 (4 KB).

Block Size Guidelines [page 85]
In nearly all cases you should rely on the default block size, which is based on the IQ page size.

5.2.1.1 Before You Create a Database

Perform prerequisite actions before creating a database using SQL statements.

In order to create a database using SQL statements, you must:

- Start the database server
- Start `dbisql`

To create a database in `dbisql`, you need to connect to an existing database, or else start the utility database, a phantom database with no database files and no data.

> **i** Note
>
> If the server is started with the `-m` server option, you cannot create a database.

In this section:

The Utility Database [page 76]
The utility database (utility_db) is a phantom database that lets you perform operations that require a connected database.

Related Information

Setting Up Windows Access to Raw Devices [page 104]

5.2.1.1.1 The Utility Database

The utility database (utility_db) is a phantom database that lets you perform operations that require a connected database.

The utility database cannot contain data because it has no database file. You can connect to the utility database in order to:

- Execute database file administration statements such as CREATE DATABASE without first connecting to an existing physical database.
- Retrieve values of connection properties and server properties.
- Connect to a running database server when it is not possible to connect to the database. For example, in a mirroring system, you can connect to the utility database to stop a mirroring server or force a database to become the primary server.

To start the utility database on a database server, specify utility_db as the database name when connecting to the server. The name utility_db is reserved. If you try to create a database with this name, the operation fails.

When connected to the utility database, you can only execute these statements:

- CREATE DATABASE statement
- DROP DATABASE statement

- CREATE USER DBA IDENTIFIED BY <new-password>
- RESTORE DATABASE statement
- REVOKE CONNECT FROM DBA
- SET TEMPORARY OPTION progress_messages = [OFF | RAW | FORMATTED]
- SELECT statement without a FROM or WHERE clause
- START DATABASE statement
- STOP DATABASE statement
- STOP SERVER statement

The `-su` database server option sets the utility database password, and optionally the user ID, or disables connections to the utility database. The default user ID for the utility database is DBA. Passwords for the utility database must be at least six characters long and are case sensitive.

The `-gu` database server option sets the privilege required for executing database file administration statements. When `-gu utility_db` is specified, only the utility database can execute the statements.

Example

Executing the following statement after connecting to the utility database creates a database named `new.db` in the directory `c:\temp`:

```
CREATE DATABASE 'c:\\temp\\new.db' DBA USER 'DBA' DBA PASSWORD 'passwd';
```

Example

Executing the following statement on the utility database returns the default collation sequence that is used when creating a database:

```
SELECT PROPERTY( 'DefaultCollation' );
```

5.2.1.2 Starting the Utility Database

You must start the utility database before creating new databases if no databases are built yet.

Context

You can start the utility database in either of these ways:

Procedure

1. Start the database server without a database by specifying only `-n enginename` on the startup command.
2. Start `dbisql` from the command line, setting the Database Name to `utility_db` in the connection string:

```
dbisql -c "uid=dba;pwd=<password>;eng=myserver;dbn=utility_db;...
```

> **i** Note
>
> Do not specify it as the Database File, because utility_db has no database file.

5.2.1.3 Dbspaces Created Automatically

When you create a database, the database server creates certain dbspaces automatically.

Dbspace Name	Purpose	Default Operating System File Name
`IQ_SYSTEM_MAIN`	Main (permanent) IQ store file	`<dbname.iq>`
`IQ_SYSTEM_MSG`	Message log file	`<dbname.iqmsg>`
`IQ_SYSTEM_TEMP`	Temporary IQ store file	`<dbname.iqtmp>`
`IQ_SHARED_TEMP`	Temporary IQ store	Initially, this dbspace contains no files.
`SYSTEM`	Catalog store file	`<dbname.db>`

The `SYSTEM` dbspace contains the system tables, which hold the schema definition as you build your database. It also holds a separate checkpoint log, rollback log, and optionally a write file, transaction log, and transaction log mirror, for the catalog store.

> **i** Note
>
> In addition to these database files, the database server also uses a temporary file to hold information needed during a session. This temporary file is not the same as the IQ temporary store, and is not needed

once the database server shuts down. The file has a server-generated name with the extension `.tmp`. Its location is determined by the TEMP environment variable, or the coordinator environment variable on UNIX.

5.2.1.4 Database File Placement

When you create a database, consider whether you will ever need to move the database. The location of files for dbspaces may also affect performance.

The IQ catalog (`.db`) and transaction log (`.log`) files can be safely moved. Never attempt to copy a running database. If you use relative path names to create the database, then you can move the files by shutting down the server and using the operating system copy file command. If you use absolute (fully qualified) path names to create the database, then you must move the files by using the `BACKUP DATABASE` command to make a full backup, and the `BACKUP DATABASE` command with the `RENAME` option to restore the backup.

IQ dbspaces on raw partitions can be moved to other partitions while the database is shut down. The new partition must be at least as large as the current dbspace size. The new partition must also have the same path in order for the dbspace to start.

You can use a byte copy utility such as dd only to copy raw data to a file system device or another raw device. Never use byte copy to move dbfiles from a file system to a raw device.

When you allocate system files for dbspaces (system, IQ main or IQ temporary), for performance reasons, SAP recommends that you not place them on a file system that is shared over a network. Any storage type can be used for SAP IQ so long as it is configured in a way that SAP IQ supports (raw devices on any platform, file systems for databases on any platform, NFS on Linux, and GPFS on Linux) and follows proper sizing guidelines.

On Windows, do not place dbspace files or log files on network drives owned by another node.

To avoid conflicts, all dbspaces should be managed by one database administrator using a single connection.

Performance related to randomly accessed files, including the system, IQ main, and IQ temporary dbfiles, can be improved by increasing the number of disk drives devoted to these files. Performance related to sequentially accessed files, including the transaction log and message log files, can be improved by locating these files on dedicated disk drives.

Suggestions to reduce file placement impact on performance:

- Keep random disk I/O away from sequential disk I/O.
- Place the database file, temporary dbspace, and transaction log file on the same physical machine as the database server.
- Isolate SAP IQ database I/O from I/O for proxy tables in other databases, such as SAP Adaptive Server Enterprise.
- Place the transaction log and message log on separate disks from the IQ store, catalog store, and temporary store, and from any proxy databases such as SAP ASE.

If your IQ requirements are large and complex enough that you need multiple physical systems, consider using SAP IQ multiplex functionality.

Raw Device Permissions

Make sure that all raw devices have read and write permissions before you create a database or add a dbspace. Check to see that `/dev/rawctl` has read permission. Raw device names on Linux use `/dev/raw/r<devname>`. For example, `/dev/raw/raw10`.

5.2.1.5 CREATE DATABASE Statement Defaults

The `CREATE DATABASE` statement has four required parameters and several optional parameters.

You must specify the default user name and password, the file name for catalog store, and the `IQ PATH`. All other parameters are optional.

If you use all of the defaults, your database has these characteristics:

- Case-sensitive (`CASE RESPECT`). “ABC” compares NOT EQUAL to “abc”. By default, passwords are case sensitive. User names are always case insensitive.
- Catalog page size of 4096 bytes (`PAGE SIZE 4096`).
- When comparing two character strings of unequal length, IQ treats the shorter one as if it were padded with blanks to the length of the longer one, so that ‘abc’ compares equal to ‘abc’ (`BLANK PADDING ON`).
- Incompatible with SAP Adaptive Server Enterprise.
- IQ page size is 128 KB (`IQ PAGE SIZE 131072`).
- IQ message file and IQ temporary store are in the same directory as the catalog store.
- For a raw device, `IQ SIZE` and `TEMPORARY SIZE` are the maximum size of the raw partition. For operating system files, see the discussion of this parameter below.
- IQ temporary store size is half the IQ size.
- jConnect JDBC driver is enabled (`JCONNECT ON`).
- The collation ISO_BINENG is used. The collation order is the same as the order of characters in the ASCII character set. In a case-sensitive database, all uppercase letters precede all lowercase letters (for example, both 'A' and 'B' precede 'a').
- `IQ RESERVE` and `TEMPORARY RESERVE` are 0.
- `SYSTEM PROCEDURE AS DEFINER` is OFF.

5.2.1.6 Increase Password Security

Passwords are an important part of any database security system. There are several options for increasing password security.

Implement a Login Policy control the frequency of password changes, to specify the number of login attempts allowed before an account is locked, or to force password expiration. See *CREATE LOGIN POLICY Statement* and *Login Policy Options* in *SAP IQ SQL Reference*.

Implement a Minimum Password Length by default, passwords can be any length. For greater security, you can enforce a minimum length requirement on all new passwords to disallow short (and therefore easily guessed) passwords. The recommended minimum length is 6. See *MIN_PASSWORD_LENGTH Option* in *SAP IQ SQL Reference*.

Implement Password Rules implement advanced password rules that include requiring certain types of characters in the password, disallowing password reuse, and expiring passwords. Validation of the rules occurs when a new user ID is created or a password is changed. See *VERIFY_PASSWORD_FUNCTION Option* in *SAP IQ SQL Reference*.

5.2.1.7 Relative Path Names

You can create a database using a relative or fully qualified path name for each of the files for the database.

Create databases with relative path names. If you specify absolute path names, you will not be able to move files to a different path name without backing up and restoring the database.

If your database is on UNIX, you can define a symbolic link for each path name.

If you omit the directory path, SAP IQ locates the files as follows:

- The catalog store is created relative to the working directory of the server.
- The IQ store, temporary store, and message log files are created in the same directory as, or relative to, the catalog store.
- The transaction log is created in the same directory as the catalog store. (This also occurs if you do not specify any file name.) However, you should place it on a different physical device from the catalog store and IQ store, on the same physical machine.

> **i** Note
>
> You must start the database server from the directory where the database is located, for any database created with a relative path name. Using a configuration file to start the server ensures that you start the server from a consistent location.

5.2.1.8 IQ PATH Parameter Guidelines

The required IQ PATH parameter tells SAP IQ that you are creating an SAP IQ database, not an SAP SQL Anywhere database.

You specify the location of your IQ store in this parameter.

Choose a location for your database carefully. Although you can move an SAP IQ database or any of its files to another location, to do so, you must shut down the database and you may have to perform a backup and restore.

You can add space on a different drive, but you can only use this additional space for new data. You cannot readily move a particular index, table, or rows of data from one location to another.

Each operating system has its own format for raw device names. See *Physical Limitations* in *SAP IQ SQL Reference* for an important note about initializing raw devices on Sun Solaris.

Raw device names on UNIX are:

- AIX – `/dev/rraw121v`
- HP-UX – `/dev/vg03/rrchee12g`

- Sun Solaris – `/dev/rsd0c`
- Sun AMD – `/dev/rdsk/c5t0d0s1`

Raw device names on Windows are:

Device Type	Name Format Required	SQL Example
Partitioned	Letter assigned to that partition	`\\\\.\\C:` (on utility database, use `\\.\C`)
Unpartitioned	`<\\?\Volume{GUID}>`. Use the Windows utility `mountvol.exe` to print the GUID of all available devices.	`\\?\Volume{f45c35fb-df13-11e7-8351-104a7de224df}`

On Windows systems, when you specify device names that include a backslash, you double the backslash to keep the system from mistaking a backslash/letter combination for an escape sequence such as tab or newline command.

To avoid problems, SAP recommends that you always double the backslash when naming raw devices on Windows in SQL statements, unless connected to the utility database. The utility database (utility_db) treats backslashes as a normal character, not an escape character..

Example 1

(Windows) The following statement creates a database called `company.db`. This database consists of four Windows files:

- The catalog store is in `company.db`, in the directory where the server was started (in this case, `c:\company`)
- The IQ store is in `c:\company\iqdata\company.iq`
- The temporary store is in `c:\company\company.iqtmp`
- The IQ message log file is in `c:\company\company.iqmsg`

```
CREATE DATABASE 'company.db'
IQ SIZE 200
IQ PATH 'c:\\company\\iqdata\\company.iq'
```

Example 2

(UNIX) The following statement creates a database called `company.db`. This database consists of four UNIX files:

- The catalog store is in `company.db`, in the directory where the server was started (in this case, `/disk1/company`)
- The IQ store is in `/disk1/company/iqdata/company.iq`
- The temporary store is in `/disk1/company/company.iqtmp`
- The IQ message log file is in `/disk1/company/company.iqmsg`

```
CREATE DATABASE 'company.db'
IQ SIZE 2000
```

```
IQ PATH '/disk1/company/iqdata/company.iq'
```

Example 3

(UNIX) The following example creates a database called `company` with a raw partition for IQ PATH:

```
CREATE DATABASE 'company'
IQ PATH '/dev/rdsk/c0t0d0s0'
```

Example 4

(Windows) The following example creates a database called `company` with a raw partition for IQ PATH:

```
CREATE DATABASE 'company'
IQ PATH '\\\\.\\D:'
```

5.2.1.9 IQ PAGE SIZE Parameter Guidelines

The IQ PAGE SIZE parameter determines memory and disk use.

You set a page size for the IQ store with the `IQ PAGE SIZE` option. The `IQ PAGE SIZE` must be a power of 2, from 65536 to 524288 bytes. The IQ page size is the same for all dbspaces in the IQ store.

To obtain the best performance, use the following minimum IQ page sizes:

- 64 KB (`IQ PAGE SIZE 65536`) for databases with the largest table contains up to 1 billion rows, or a total size less than 8 TB. This is the absolute minimum for a new database. On 32-bit platforms, a 64 KB IQ page size gives the best performance.
- 128 KB (`IQ PAGE SIZE 131072`) for databases on a 64-bit platform with the largest table contains more than 1 billion rows and fewer than 4 billion rows, or may grow to a total size of 8TB or greater. 128 KB is the default IQ page size.
- 256 KB (`IQ PAGE SIZE 262144`) for databases on a 64-bit platform with the largest table contains more than 4 billion rows, or may grow to a total size of 8 TB or greater.

Multiuser environments, and systems with memory constraints, both benefit from an IQ page size of at least 64 KB, as this size minimizes paging.

SAP IQ stores data on disk in compressed form. It uncompresses the data and moves data pages into memory for processing. The IQ page size determines the amount of disk compression and the default I/O transfer block size for the IQ store. For most applications, this default value is best. For information on these settings and other options that affect resource use and performance, see *SAP IQ Performance and Tuning Guide*.

Page Size for Wide Data

If your database includes very wide tables, you may find that the next higher IQ page size for a given number of rows gives you better performance. For example, tables with multiple columns of wide `CHAR` or `VARCHAR` data (columns from 255 to 32,767 bytes) are likely to need a larger than usual IQ page size.

Because IQ stores data in columns, it does not have a true maximum row length. The practical limit, however, is half your IQ page size, because that is the widest result set that a query is guaranteed to be able to return to the client. Choose an IQ page size at least twice the width of the widest table possible.

5.2.1.10 Database Size Guidelines

When you create a database, you set the size and reserve size in MB of the initial database file (the `IQ_SYSTEM_MAIN` dbspace).

These values are defined in the `IQ SIZE` and `IQ RESERVE` parameters for the main store and `TEMPORARY SIZE` and `TEMPORARY RESERVE` for the temporary store.

- For raw partitions, you do not need to specify `IQ SIZE` or `TEMPORARY SIZE`; SAP IQ determines the size of the raw devices and sets `IQ SIZE` and `TEMPORARY SIZE` automatically. If you do specify size, the size cannot be larger than the actual raw partition size.
- For operating system files you can rely on the defaults or specify a value based on the size of your data, from the required minimums up to a maximum of 4 TB, in 1 MB increments.

In this section:

Default and Minimum Sizes of IQ and Temporary Stores [page 84]
The `IQ RESERVE` and `TEMPORARY RESERVE` parameters reserve a range of blocks, so that the dbspace can be resized at a later time. Making `IQ RESERVE` larger than needed can use additional disk space, however.

5.2.1.10.1 Default and Minimum Sizes of IQ and Temporary Stores

The `IQ RESERVE` and `TEMPORARY RESERVE` parameters reserve a range of blocks, so that the dbspace can be resized at a later time. Making `IQ RESERVE` larger than needed can use additional disk space, however.

	Default Size		Minimum Size When Specified Explicitly	
IQ Page Size	**IQ Store**	**Temporary Store**	**IQ Store**	**Temporary Store**
65536	4096000	2048000	4 MB	2 MB
131072	8192000	4096000	8 MB	4 MB
262144	16384000	8192000	16 MB	8 MB
524288	32768000	16384000	32 MB	16 MB

5.2.1.11 PAGE SIZE Parameter Guidelines

Set the catalog store page size with the `CREATE DATABASE PAGE SIZE` option. The default and minimum value for this option is 4096 (4 KB).

Example

The following statement creates a database with a catalog `PAGE SIZE` of 4096 bytes (4 KB), where the IQ store is on a UNIX raw partition and has an `IQ PAGE SIZE` of 131072 bytes (128 KB):

```
CREATE DATABASE 'company'
IQ PATH '/dev/rdsk/c2t6d0s3'
PAGE SIZE 4096
IQ PAGE SIZE 131072
```

By default, the IQ store size is the size of the raw partition and the temporary store is half that size. Because no path is specified for the temporary store, it is created in the same directory as the catalog store.

5.2.1.12 Block Size Guidelines

In nearly all cases you should rely on the default block size, which is based on the IQ page size.

5.2.2 IQ Main Store and IQ Temporary Store Space Management

Options `MAIN_RESERVED_DBSPACE_MB` and `TEMP_RESERVED_DBSPACE_MB` provide room for checkpoint, commit, and release savepoint operations.

These options determine the reserve space allocation size in the last readwrite dbfile in IQ_SYSTEM_MAIN or IQ_SYSTEM_TEMP, respectively.

The user with the SET ANY PUBLIC OPTION system privilege can limit the amount of space used per connection. In addition, when SAP IQ runs out of space in IQ main store or the IQ temporary store, the server no longer suspends the transaction that ran out of space until new space is added. The transaction that runs out of space in the IQ main store or the IQ temporary store fails and is rolled back.

The database option `MAX_TEMP_SPACE_PER_CONNECTION` limits the amount of IQ temporary store space used per connection and tracks temporary store usage for all Data Manipulation Language (DML) statements, in addition to queries. `MAX_TEMP_SPACE_PER_CONNECTION` monitors and limits the actual run time temporary store usage by the statement. If the connection exceeds the quota set by the `MAX_TEMP_SPACE_PER_CONNECTION` option, an error is returned and the current statement rolls back.

The default value of the `QUERY_TEMP_SPACE_LIMIT` database option is 0, which means there is no limit on temporary store usage by queries. To limit the temporary store usage per connection, the DBA can set the `MAX_TEMP_SPACE_PER_CONNECTION` option for all DML statements, including queries.

When an SAP IQ database is upgraded from a release prior to version 15.0, the `MAX_TEMP_SPACE_PER_CONNECTION` database option is set to the default value of 0. You can use `sp_iqcheckoptions` to find the default and current values of options before and after upgrading, to help determine if the new option settings are appropriate for the upgraded database.

5.3 Dropping a Database

Dropping a database deletes all tables and data from disk, including the transaction log that records alterations to the database. It also drops all of the dbspaces associated with the database.

Procedure

To drop a database, use the following SQL statement:

```
DROP DATABASE <dbname>
```

You must specify the database name and its path name exactly as they were specified when the database was created.

For example, on a Windows system:

```
DROP DATABASE 'c:\sap\data\mydb.db'
```

You cannot drop the database until you stop it. If the connection parameter `AUTOSTOP=no` is used, you may need to issue a `STOP DATABASE` statement.

5.4 Schema and Data Export

The `iqunload` utility lets you make a full or partial copy of an existing database schema and then load it into another existing database of the same installation version. Data from the original database can be optionally included.

`iqunload` creates an Interactive SQL script file. If data is included, all data in each specified table is copied into files in the specified directory, in comma-delimited format. Binary data is properly represented with escape sequences.

The dbo user ID owns a set of system objects in a database, including views and stored procedures. The `iqunload` utility does not unload the objects that were created for the dbo user ID during database creation. Changes made to these objects, such as redefining a system procedure, are lost when the database is unloaded. Any objects that were created by the dbo user ID since the initialization of the database are unloaded by the Unload utility, and so these objects are preserved. When you unload a database, changes to permissions on system objects are not unloaded. You must grant or revoke these permissions in the new database.

In this section:

iqunload Schema and Data Export Utility [page 87]
A command line utility to unload a full or partial schema based on a subset of tables and users. Data can optionally be included in the unload.

Load Schema and Data [page 91]
Run the SQL script generated by the unload to load the schema and any data into a new database.

5.4.1 iqunload Schema and Data Export Utility

A command line utility to unload a full or partial schema based on a subset of tables and users. Data can optionally be included in the unload.

Run `iqunload` against an SAP IQ 15.4 or later database to unload the database schema, and optionally, its data into a SQL script file. You can then load the schema file into another SAP IQ database of the same version.

> **i** Note
>
> When you run `iqunload` against an SAP IQ 12.7 database, `iqunload` instead migrates the database, re-creating the legacy catalog on the new database catalog in the current installation. To use `iqunload` to migrate a SAP IQ 12.7 database, see *iqunload Utility* in the *SAP IQ Installation and Update Guide* documentation for your platform.

See the *SAP IQ Administration: Database* documentation for details on exporting schema.

> Syntax

```
iqunload [ <parameters> ]  <directory> [ @<data> ]
```

Parameters

parameters

-c "keyword = value, ..."

Supplies database connection parameters. Use the DBF parameter to specify the name of the database file for unload. The file path is either absolute or relative to the server startup directory.

For example, the following statement unloads the database `mydemo`, connecting as user `DBA` with password `sql`. The data is unloaded into the `c:\unload` directory:

```
iqunload -c "DBF=c:\IQdemo\mydemo.db;UID=DBA;PWD=<password>" c:\unload
```

-dc

(Optional) forces all computed columns in the database to be recalculated.

-n

Unloads the schema definition only; no data.

-null_string text

(Optional) applies to tables containing a varchar or varbinary data types only. During unload, provides a means to differentiate between NULL values and string values that contain the word NULL. The value of `-null_string` should not match any other pattern of values in the table.

If not defined, defaults to NULL.

-o messages_file_name

(Optional) logs output messages to the named file. The location of this file is relative to `iqunload`.

-q

(Optional) suppresses messages and windows.

> **i** Note
>
> When using `-q`, also specify `-y` or `iqunload` fails if the generated SQL script file already exists.

-r script_file_name

(Optional) specifies the name and directory of the generated SQL script file. The default is `reload.sq` in the current directory.

-t table_list

(Optional) outputs listed tables only. Identify tables in a comma separate list as `<OwnerName>.<TableName>` or `<TableName>` alone.

> **i** Note
>
> Cannot be used with `-tl` parameter.

-tl 'table_pattern1,table_pattern2,...'

(Optional) unloads tables matching the specified pattern, listed in a comma separate list.

When used in conjunction with `-ul`, if a specified table is owned by a user not included in the `-ul` list, the table is not unloaded.

When used in conjunction with `-tld`, if the same table is not specified by both clauses, only the schema for the specified table is unloaded. For example, `-tl table1, table2 -tld table1` unloads the schema and data for `table1`, but the schema only for `table2`.

Cannot be used with `-t` parameter.

When `-n` not specified, all data for specified tables is unloaded.

-tld 'table_pattern1:condtion1, table_pattern2:condition2'

(Optional) restricts the unloaded of data matching the specified pattern within tables. List tables with conditions in a comma separated list.

To further restrict the data unload within the table, use a colon (:) followed by a `where` clause. All data in the specified table is unloaded if no `where` clause is specified.

When used with `-tl`, if the same table is not specified by both clauses, no data is unloaded for the specified table. For example, `-tl table1 -tld table1, table2` unloads the schema and data for `table1`, but no schema or data for to running the utility.`table2`.

When `-tl` not specified, all tables are considered for unload, then conditions apply., but no schema or data for

When `-n` is also specified, no data is unloaded regardless of conditions.

-ul 'user_pattern1, userpattern2, ...'

(Optional) unloads objects owned by users matching the specified comma separate list.

-up

(Optional) unloads encrypted passwords.

-v

(Optional) displays the name of the table being unloaded, and the number of rows that have been unloaded.

-y

(Optional) replaces existing schema SQL script with new output without confirmation.

> **i** Note
>
> When using `-q`, also specify `-y` or `iqunload` fails if the generated SQL script file already exists.

directory

Specifies the directory where the data files referenced in the unloaded

> **i** Note
>
> Not required when used with `-n`.

@data

Reads in options from the specified environment variable or configuration file.

Remarks

The generated schema must be loaded into a database of the same version as the database from which was unloaded.

The `-tl`, `-tld`, and `-ul` parameters support pattern matching to define a subset of tables and users for unload. The pattern list is read iteratively from left to right, building a temporary list of objects to unload. If the first character of a pattern is not an apostrophe (!), all objects matching the pattern are added to the temporary list. If the first character of the pattern is an apostrophe, any object on the temporary list matching the pattern is removed from the list.

For example, when building the temporary list, the syntax `-tl '!table1,table1'` would first remove all tables named table1 from the temporary list and then re-add all tables named `table1`. The net result is only tables named `table1` are unloaded. Conversely, the syntax `-tl 'table1,!table1'` first limits the temporary list to only tables named `table1` and then removes all tables named `table1` from the list. The net result now is that no tables are unloaded.

Pattern-matching uses `SQL LIKE` syntax (%, *, ?, etc.)

If no `-ul` parameter is specified, there are no restrictions on owners for tables unloaded.

When `-n` is specified, no data unloaded regardless of other clauses. When `-n` is not specified, the data unloaded is dependent on other clauses specified.

To exclude users or tables in an `IQUNLOAD` statement, you must first specify the pool of users or tables to be considered for the exclusion. For example, to exclude tables beginning with the letter `t` (`!t%`), you must first include all tables (`%`): `-tl '%,!t%'`.

Permissions

Requires DBA authority for 15.4 databases.

Requires the SELECT ANY TABLE system privilege for 16.0 and later databases.

Examples

- This example exports the database schema, and exports encrypted passwords:

  ```
  iqunload -n -c "uid=DBA;pwd=*****;eng=iq_iqdemo;dbn=iqdemo;" -up -r
  iqdemo_unload.sql
  ```

  > **i** Note
  >
  > If you do not specify the `-up` option, then all `GRANT CONNECT TO <user>` commands are printed without the `IDENTIFY BY` clause in the generated file. This means users do not have a password, and cannot connect to the database.

- This example unloads all tables beginning with the letter `t`:

  ```
  iqunload -n -c "..." -tl 't%'
  ```

- This example unloads all tables not beginning with the letter `t`:

  ```
  iqunload -n -c "..." -tl '%,!t%'
  ```

- This example unloads all table named `table1`, or table owned by user `user1` that begins with the letter `t`, excluding any table named `t2`:

  ```
  iqunload -n -c "..." -tl 'table1,user1.t%,!t2'
  ```

- This example unloads any table beginning with the letters `ta` and owned by a user name beginning with `sm`:

  ```
  iqunload -n -c "..." -tl 'sm%.ta%'
  ```

- This example unloads any table owned by a user name containing the letters `em` but not equal to `temp`:

  ```
  iqunload -n -c "..." -ul '%em%,!temp'
  ```

  If tables `u1.table1` and `u2.table1` exist, both tables are unloaded by this syntax:

  ```
  iqunload -n -c "..." -tl 'table1'
  ```

But only `u1.table1` is unloaded if the owner is included in the syntax:

```
iqunload -n -c "..." -tl 'u1.table1'
```

- This example unloads only rows for `table1` where `column c_char1` is not null. For `table2`, only rows where vb is null are unloaded. For all other tables (`%`), all data is unloaded. The data is unload to the directory `./data` relative to the server, in a file named `data_reload.sql`:

  ```
  iqunload -c "..." -tld '%,table1:c_char1 is not null, table2:vb is null'-r
  data_reload.sql ./data
  ```

- This example unloads schema for all tables of the form `ta%`. It unloads all data for tables `table1`, but only those rows in `table6` where `c1 < 0`. The data is unload to the directory `./data` relative to the server, in a file named `data_reload.sql`:

  ```
  iqunload -c "..." -tl 'ta%' -tld '%.ta%:none, table1, table6:c1 < 0, !table3'
  -r data_reload.sql ./data
  ```

- This example unloads schema and data for `table1` and uses `NULL_NULL` for `null_string`. The data is unload to the directory `./data` relative to the server, in a file named `data_reload.sql`.

  ```
  iqunload -c "..." -tl 'table1' -null_string NULL_NULL -r data_reload.sql ./
  data
  ```

5.4.2 Load Schema and Data

Run the SQL script generated by the unload to load the schema and any data into a new database.

Prerequisites

- You have the SERVER OPERATOR system privilege in the new database.
- The new database already exists.
- On the host of the new database, the unload data files reside in the same relative path as on the original host.

Context

The data files created during unload are referenced by the reload SQL script. If the data files cannot be found by the script, the reload fails. If a data file is empty (the specified table was empty), a warning message appears, but the table schema is loaded, and the process continues.

Procedure

1. Connect to the new database.
2. Open the SQL script and run it.

5.5 Disconnecting Other Users from a Database

SAP IQ gives you the ability to disconnect other users from a given database.

Context

You can obtain the `<connection-id>` for a user by using the `connection_property` function to request the connection number. The following statement returns the connection ID of the current connection:

```
SELECT connection_property( 'number' )
```

Procedure

1. Connect to an existing database as a user with administrative authority over the DROP CONNECTION system privilege.
2. Using Interactive SQL, execute a `DROP CONNECTION` statement.

   The following statement drops the connection with ID number 4.

   ```
   DROP CONNECTION 4
   ```

5.6 Disconnecting from a Database in Embedded SQL

When you are finished working with a database, you can disconnect a named connection or all connections.

Procedure

Execute an `EXEC SQL DISCONNECT` statement.

The following statement shows how to use DISCONNECT in Embedded SQL:

```
EXEC SQL DISCONNECT :conn_name
```

5.7 Disconnecting All Connections from a Database in Interactive SQL

When you are finished working with a database, you can disconnect a named connection or all connections.

Procedure

Execute a `DISCONNECT` statement.

The following statement shows how to use `DISCONNECT` from Interactive SQL to disconnect all connections:

```
DISCONNECT ALL
```

5.8 Showing System Objects in Interactive SQL

In Interactive SQL, you cannot query system tables, but you can browse the contents of a system view.

Context

Most system tables have equivalent system views that you can query.

In a database, a table, view, stored procedure, or domain is a system object. System tables store information about the database itself, while system procedures, and domains largely support SAP Transact-SQL compatibility.

Procedure

1. Connect to a database using Interactive SQL.
2. Execute a `SELECT` statement, specifying the system view for the table you want to browse.

   To browse the `ISYSTAB` system table, show the contents of the view `SYS.SYSTAB` in the Results pane.

   ```
   SELECT *
   ```

```
FROM SYS.SYSTAB
```

5.9 Setting Database Options in Interactive SQL

Database options are configurable settings that change the way the database behaves or performs.

Procedure

Specify the properties within a SET OPTION statement.

> **i** Note
>
> When you set options for the database itself, you are actually setting options for the PUBLIC role in that database, because all users and roles inherit option settings from PUBLIC.

6 Manage Data Storage in SAP IQ

Specify how SAP IQ allocates disk space and organizes data on disk for optimal performance.

In a production environment, raw partition installations may improve processing performance and recovery capabilities. File systems, however, offer easier device management, and may be preferable in a development environment.

In this section:

Space Allocation [page 96]
All SAP IQ databases are preallocated, whether they reside in a file system or a raw partition.

Types of Dbspaces [page 96]
Each type of dbspace stores a particular type of SAP IQ data.

Space for Databases [page 101]
The first dbspace for each store is created automatically when you create the database. Create additional dbspaces as needed.

Space Requirements for IQ Stores and Temporary Stores [page 102]
The amount of data, and the number and types of indexes you create, determine how much space you need in your database.

Sizing Guidelines for Main and Temporary Stores [page 102]
The SAP IQ architecture influences guidelines for data storage.

Setting Up Windows Access to Raw Devices [page 104]
Windows systems restrict raw device access to user accounts with Administrator privilege.

Choosing Drives for Raw Devices on Windows [page 105]
Some available drives may be inappropriate for raw devices.

Viewing Access Permissions of a Raw Device [page 106]
Display the current access permissions of a Windows raw device.

Setting Permissions for a Raw Device [page 107]
You must set access permissions for a raw device on Windows before you can use it.

Estimating Space and Dbspaces Required [page 108]
To avoid difficulties when a database or a particular dbspace is full, estimate dbspace requirements before you create the database and the objects in it.

Dbspace Management Example [page 109]
This section illustrates the dbspace management process from creating a new database and adding objects and data to the database, through relocating objects and dropping the empty dbspace. This example includes sample SQL code and the output of the related system stored procedures.

Data Storage [page 114]
A DBA can determine which tables and indexes reside on a given dbspace, relocate objects to other dbspaces, and drop any dbspace after emptying it of data.

Cache Dbspace for Direct-Attached Storage [page 126]

Add direct-attached storage devices, such as solid-state disks (SSDs), to a multiplex node or a simplex server by configuring a cache dbspace local to each node. Local direct-attached storage provides an SSD buffer pool extension to your database, which can improve I/O performance in general, and shared SAN device performance in particular. Although multiplex configurations gain the most benefit from direct-attached storage, simplex servers can benefit too.

6.1 Space Allocation

All SAP IQ databases are preallocated, whether they reside in a file system or a raw partition.

Each database includes multiple tablespaces. A tablespace is a unit of storage within the database that you can administer as a logical subset of total storage. You can allocate individual objects and subobjects to individual tablespaces.

A dbspace is a tablespace that consists of one or more operating system files.

A store is one or more dbspaces that store persistent or temporary data for a special purpose. SAP IQ has five stores:

- The catalog store contains the SYSTEM dbspace and additional user-defined catalog dbspaces.
- The IQ main store contains the IQ_SYSTEM_MAIN dbspace and other user dbspaces.
- The IQ temporary store contains the IQ_SYSTEM_TEMP dbspace.
- The IQ shared temporary store contains the IQ_SHARED_TEMP dbspace.
- The row-level versioning (RLV) store is an in-memory store for high-performance row-level updates.

6.2 Types of Dbspaces

Each type of dbspace stores a particular type of SAP IQ data.

Dbspace Type	Data Stored	Files Contained by Dbspace	Number of Dbspaces
SYSTEM dbspace	System tables, views, stored procedures, SAP SQL Anywhere tables, and function definitions.	One	One or more
Other catalog dbspaces	SAP SQL Anywhere tables.	One	One or more

Dbspace Type	Data Stored	Files Contained by Dbspace	Number of Dbspaces
IQ_SYSTEM_MAIN	Database structures including rollforward and rollback data for each committed transaction and each active checkpointed transaction, the incremental backup metadata, and database space and identity metadata. You may store user objects here, but it is better to store them in other main dbspaces.	One or more	One
Other main dbspaces (also called user dbspaces)	Database objects such as tables, indexes, and table metadata.	One or more	One or more. You must be licensed for the IQ_VLDBMGMT option to create more than one user dbspace.
RLV_STORE	Real-time in-memory data from row-level versioning (RLV)-enabled tables.	One	One
IQ_SYSTEM_TEMP	Set of 1 to n temporary dbfiles that defines a single temporary dbspace for a standalone database or multiplex node. See the following table for details.	One or more	One
IQ_SHARED_TEMP	Set of 1 to n temporary dbfiles that define a single temporary dbspace shared by all multiplex nodes. See the following table for details.	One or more (initially has no files)	One
IQ_SYSTEM_MSG	External file that logs messages about database activity.	One per multiplex node	One
Cache dbspace	An independent dbspace, like the IQ main store, IQ temporary store, and RLV dbspaces, the cache dbspace can contain a single file or device, or multiple files or devices.	One or more	

The dbspace of a table is implicitly or explicitly specified. For base tables, the value of the `DEFAULT_DBSPACE` option implicitly determines the dbspace location, or you can explicitly specify the location using the `CREATE TABLE IN <dbspace_name>` clause. Typically, create base tables in a dbspace in the IQ main store, but you can also create them without IQ indexes in a dbspace in the catalog store.

For global temporary tables, specify the `IN SYSTEM` clause to explicitly create an SA global temporary table. By default, IQ temporary tables are created in IQ_SYSTEM_TEMP.

Always Stored in IQ_SYSTEM_TEMP	Always Stored in IQ_SHARED_TEMP	May Be Stored in Either IQ_SHARED_TEMP or IQ_SYSTEM_TEMP
Scratch space for recovery from server failure	Distributed temporary objects for distributed query processing	IQ temporary user objects such as IQ temporary tables and indexes
Local bitmaps for shared storage space management		Scratch space for local, nonversioning temporary objects
Temporary storage transaction state (savepoint roll forward and roll back)		

In this section:

Catalog Store [page 98]
The catalog store contains the metadata for the SAP IQ database.

IQ_SYSTEM_MAIN Dbspace [page 99]
IQ_SYSTEM_MAIN is a special dbspace that contains structures necessary for the database to open: the checkpoint log, rollforward/rollback data for each committed transaction and each active checkpointed transaction, the incremental backup metadata, and database space and identity metadata.

Other User Main Dbspaces [page 99]
Create user main dbspaces so that users do not place user tables or indexes in IQ_SYSTEM_MAIN.

IQ Temporary Dbspace [page 99]
A single dbspace for the IQ temporary store, IQ_SYSTEM_TEMP, is created when you create or upgrade a database.

IQ Shared Temporary Dbspace [page 100]
A single dbspace for the IQ shared temporary store, IQ_SHARED_TEMP, is created when you create or upgrade a database to SAP IQ 16.1. This dbspace stores temporary structures that are shared among nodes for distributed query processing.

IQ Message File Dbspace [page 101]
IQ_SYSTEM_MSG is a system dbspace that points to the file path of the database IQ message log file.

6.2.1 Catalog Store

The catalog store contains the metadata for the SAP IQ database.

Metadata describes the layout of tables, columns, and indexes. The catalog store is sometimes referred to simply as the catalog.

- The SYSTEM dbspace – The catalog dbspace SYSTEM contains metadata for your database, stored in the same format as tables in an SAP SQL Anywhere relational database system. SAP SQL Anywhere can exist with or without SAP IQ. You may have SAP SQL Anywhere-style tables in your catalog store along with your SAP IQ tables, or you may have a separate SAP SQL Anywhere database. Each catalog dbspace contains exactly one file.
- Other catalog dbspaces – you can create SAP SQL Anywhere tables in a separate dbspace from the SYSTEM dbspace.

You can preallocate space to your catalog store by executing an `ALTER DBSPACE` statement.

> **i** Note
>
> The catalog store cannot be on a raw partition.

6.2.2 IQ_SYSTEM_MAIN Dbspace

IQ_SYSTEM_MAIN is a special dbspace that contains structures necessary for the database to open: the checkpoint log, rollforward/rollback data for each committed transaction and each active checkpointed transaction, the incremental backup metadata, and database space and identity metadata.

The IQ_SYSTEM_MAIN dbspace is created at database creation or when you upgrade an older database to SAP IQ 16.1. IQ_SYSTEM_MAIN is always online when the database is open.

6.2.3 Other User Main Dbspaces

Create user main dbspaces so that users do not place user tables or indexes in IQ_SYSTEM_MAIN.

The best practice is to avoid placing user tables or indexes in IQ_SYSTEM_MAIN. The administrator may allow user tables to be created in IQ_SYSTEM_MAIN, especially if these tables are small, very important tables. However, the recommended method is that immediately after creating the database, the administrator creates a second main dbspace (a user main dbspace), revokes CREATE privilege in dbspace IQ_SYSTEM_MAIN from PUBLIC, grants CREATE privilege for the new main dbspace to selected users or PUBLIC, and sets PUBLIC.DEFAULT_DBSPACE to the new user main dbspace.

For example:

```
CREATE DBSPACE user_main USING FILE user_main
'user_main1' SIZE 10000;
GRANT CREATE ON user_main TO PUBLIC;
REVOKE CREATE ON IQ_SYSTEM_MAIN FROM PUBLIC;
SET OPTION PUBLIC.DEFAULT_DBSPACE = 'user_main';
```

> **i** Note
>
> You must be licensed for the IQ_VLDBMGMT option to create more than one user dbspace.

6.2.4 IQ Temporary Dbspace

A single dbspace for the IQ temporary store, IQ_SYSTEM_TEMP, is created when you create or upgrade a database.

Each dbspace may contain any number of files. The only limit is that the total number of IQ files is 16383.

Ensure that the IQ_SYSTEM_TEMP dbspace is at least as large as the value specified in the `-iqtc` server option of the `start_iq` utility.

6.2.5 IQ Shared Temporary Dbspace

A single dbspace for the IQ shared temporary store, IQ_SHARED_TEMP, is created when you create or upgrade a database to SAP IQ 16.1. This dbspace stores temporary structures that are shared among nodes for distributed query processing.

Initially, this dbspace contains no files. To add files, use `ALTER DBSPACE ADD FILE`. Allocating files to this dbspace is optional and required only for distributed query processing in multiplex servers.

When you set the `TEMP_DATA_IN_SHARED_TEMP` logical server policy option ON, all temporary table data and eligible scratch data writes to the shared temporary store, provided that the shared temporary store is not empty. You must restart secondary nodes after setting this option or after adding a read-write file to the shared temporary store. If the shared temporary store contains no read-write file, or if you do not restart secondary nodes, data instead writes to IQ_SYSTEM_TEMP. When OFF, all temporary table data and scratch data writes to the local temporary store.

Shared System Temporary Store

A multiplex configuration with shared temporary storage can use the IQ_SHARED_TEMP dbspace as a shared system temporary store instead of requiring a separate local store for each secondary server. The shared system temporary store simplifies multiplex configuration, improves performance, and supports distributed query processing.

> **i Note**
>
> Do not use the IQ_SHARED_TEMP dbspace as a local store. See SAP Note 2453117.

On multiplex systems:

- When you set the logical server policy option TEMP_DATA_IN_SHARED_TEMP ON, SAP IQ creates all temporary objects on the IQ_SHARED_TEMP dbspace. You must restart secondary nodes after setting this option or after adding a read-write file to the shared temporary store. (If the shared temporary store contains no read-write file, or if you do not restart secondary nodes, data instead writes to IQ_SYSTEM_TEMP.)
- Temporary user objects (such as tables or table indexes) that you create using the `IN IQ_SYSTEM_TEMP` clause go in either IQ_SYSTEM_TEMP or IQ_SHARED_TEMP, depending on the value of the logical server option TEMP_DATA_IN_SHARED_TEMP:
  - If TEMP_DATA_IN_SHARED_TEMP is 'OFF', objects go in IQ_SYSTEM_TEMP.
  - If TEMP_DATA_IN_SHARED_TEMP is set 'ON', objects go in IQ_SHARED_TEMP.

  SAP IQ does not support creating temporary user objects using the `IN IQ_SHARED_TEMP` clause.
- The `WITH STOP SERVER` clause automatically shuts down all servers in the logical server. These statements support `WITH STOP SERVER`:
  - `ALTER LOGICAL SERVER`

- `ALTER LS POLICY`
- `CREATE LOGICAL SERVER`
- `DROP LOGICAL SERVER`

See *SAP IQ SQL Reference* for complete syntax information on these statements.

- If you use `ALTER LS POLICY ... WITH STOP SERVER` to change the TEMP_DATA_IN_SHARED_TEMP option 'ON | OFF', all servers in that logical server shut down automatically. You must restart the servers to force the logical server to place temporary data in the store specified by the TEMP_DATA_IN_SHARED_TEMP option.
- If you use `ALTER LS POLICY` to set TEMP_DATA_IN_SHARED_TEMP 'OFF', the logical server starts placing temporary data in the SYSTEM temporary area after the next normal server startup.
- You can also change the TEMP_DATA_IN_SHARED_TEMP value indirectly using `CREATE LOGICAL SERVER`, `ALTER LOGICAL SERVER`, or `DROP LOGICAL SERVER` statements and the `WITH STOP SERVER` clause.

6.2.6 IQ Message File Dbspace

IQ_SYSTEM_MSG is a system dbspace that points to the file path of the database IQ message log file.

IQ_SYSTEM_MSG is not considered a store because it does not store any data. It has one file per multiplex node. By default, the physical file name for the message file on an SAP IQ server or a coordinator of a multiplex is `<dbname>.iqmsg`. The physical file name for the IQ message file on a secondary node in a multiplex is `<servername>.iqmsg`.

Because it is not an IQ store dbspace, `ALTER` commands such as `READONLY` and `OFFLINE` do not apply to IQ_SYSTEM_MSG.

6.3 Space for Databases

The first dbspace for each store is created automatically when you create the database. Create additional dbspaces as needed.

When you create and load a table, SAP IQ distributes data among all existing dbspaces in that store with available space. You can reserve space for a dbspace to grow when you create it. You can resize the dbspace up to the maximum reserve. You can also make the dbspace smaller, provided that all data has been moved off of the truncated portion of the dbspace. You can move individual database objects off of specified dbspaces as needed.

Do not allocate all of your disk space to your database. Keep ten percent in reserve. SAP IQ needs this space to gracefully handle out-of-space conditions.

Create all dbspaces when you create the database, rather than adding them gradually as old ones fill. This ensures that your dbspaces are filled more evenly, improving disk I/O.

Create separate databases for debugging purposes. Avoid performing development work on production databases, because it increases the possibility of a server failure.

6.4 Space Requirements for IQ Stores and Temporary Stores

The amount of data, and the number and types of indexes you create, determine how much space you need in your database.

If you run out of space when loading or inserting into a database, the server rolls back either the entire transaction or rolls back to a savepoint.

In addition to any temporary tables you explicitly define, SAP IQ uses the temporary store as a temporary result space for sorts, hashes, and bitmaps during loads and deletions. The types of queries issued, the degree of concurrent use, and the size of your data all determine how much space you need for your temporary store.

6.5 Sizing Guidelines for Main and Temporary Stores

The SAP IQ architecture influences guidelines for data storage.

- The IQ_SYSTEM_MAIN dbspace holds all of the database metadata other than IQ table metadata. IQ table metadata is stored in the table's dbspace and the table version (TLV) log. If a node is down, the multiplex must store versions to synchronize them when the node comes back up. These versions may use large amounts of space.
- Approximately 20 percent of the IQ_SYSTEM_MAIN dbspace is now used for preallocated free list space and not available for user data.
- This version of SAP IQ performs more operations in parallel, so it uses more temporary space than earlier versions.

Three factors influence the space required for the IQ_SYSTEM_MAIN store:

- Versioning – the volume of versions maintained varies.
- Nature of data and indexes.
- Dynamic nature of the data – the capacity to load more data at any time.

While documentation can offer general guidelines, the combination of these factors makes each database's requirements unique. For a development or report server with a total size under 500 GB, an IQ_SYSTEM_MAIN file of 10 to 20 GB may suffice.

Task	Guideline	Notes
Loading empty schema from `iqunload -n` output or for a small test database	10 GB main, 5 GB temporary	CREATE DATABASE sizes are in MB.

Task	Guideline	Notes
Creating new production database	• If you are migrating a database, and use a raw device for your current IQ_SYSTEM_MAIN, assign a new unused raw device of your standard size. • Total size of IQ_SYSTEM_MAIN should be at least 1/100 total database size, with a minimum 100 GB main and 100 GB reserve. • If using raw disks for IQ_SYSTEM_MAIN, use multiple raw disks whenever possible. Multiple raw disks enable SAP IQ to stripe the data across devices, which improves performance. • For IQ dbspaces in production, use a fault-tolerant file system implemented by a high-performance, redundant disk array (for example, RAID 5). For single-server systems, you can use a local file system, but multiplex systems require raw devices, ideally on a Storage Area Network device.	Omit `ms_size` if specifying a raw device. On a Windows system, only user accounts with Administrator privilege can access raw devices. The `rawaccedit` utility sets permission for devices for the current session. Always set the main reserve to 20 percent of IQ_SYSTEM_MAIN size. To set the main reserve, use the database option `MAIN_RESERVED_DBSPACE_MB`.
Creating main store for a multiplex	Double the space recommended for a simplex database, or at least 200 GB main and 200 GB reserve dbspace.	

Example 1

In `CREATE DATABASE` syntax, default size units are in MB, not GB.

The following statement creates a database with 100 GB IQ_SYSTEM_MAIN with 100 GB reserve (for future expansion):

```
CREATE DATABASE 'test.db'
                    IQ PATH 'test.iq'
                    IQ SIZE 100000
                    IQ RESERVE 100000
                    TEMPORARY PATH 'test.iqtmp'
                    TEMPORARY SIZE 5000
```

Example 2

`MAIN_RESERVED_DBSPACE_MB` lets you control the amount of space SAP IQ sets aside in your IQ main store for certain small but critical data structures used during release savepoint, commit, and checkpoint operations.

Set the `MAIN_RESERVED_DBSPACE_MB` option value to 20 percent of the IQ_SYSTEM_MAIN SIZE. For example, if IQ_SYSTEM_MAIN is 100 GB, set it to 20 GB, as follows:

```
SET OPTION PUBLIC.MAIN_RESERVED_DBSPACE_MB = 20000
```

Example 3

Specify the IQ_SYSTEM_MAIN size in the database migration command.

The `-ms_size` parameter requires a value in MB, not GB. Omit `-ms_size` if specifying a raw device. For a raw device, you must specify an unused raw partition. The following creates an IQ_SYSTEM_MAIN on a raw device:

- On UNIX:

```
iqunload -au -ms_filename /dev/rdsk/c1t0d1 -c
"UID=DBA;PWD=<password>;DBF=latest.db"
```

- On Windows:

```
iqunload -au -ms_filename \\\\.\\PhysicalDrive1 -c
"UID=DBA;PWD=<password>;DBF=latest.db"
```

6.6 Setting Up Windows Access to Raw Devices

Windows systems restrict raw device access to user accounts with Administrator privilege.

Context

To run the SAP IQ servers using an account that lacks Administrator privilege, you must enable new device access permissions for that account after each system reboot.

The `rawaccedit` utility sets permissions for devices for the current session.

Set up read-write access for the write servers and read access for query servers.

Procedure

1. Type the following at a command prompt:

```
rawaccedit
```

2. In the IQ Raw Device Access window, type the name of the user and the device to which you want to grant access.

   You can use `Alt` + `n` to tab to the User's Name box and `Alt` + `d` to tab to the Raw Device Name box.

Option	Description
To specify an unpartitioned raw device	Type unpartitioned drives are named with the format `\\?\Volume{GUID}`, where `<GUID>` is the device identifier. To find the device identifier, you may use the Windows utility `mountvol.exe` to print the GUID of all available devices.
To specify a partitioned raw device	Type the letter assigned to that partition.

3. Click **Add**.
4. Correct any errors in the user name and device name that display in the top panel and click **Update ACL and Exit**.

Results

Device access permissions remain until you restart Windows.

6.7 Choosing Drives for Raw Devices on Windows

Some available drives may be inappropriate for raw devices.

Procedure

1. Open **My Computer** or **Computer** to see available logical drives.

   On Windows, raw devices can be named using logical drive letters or the physical disk name.

   Local disks that are formatted list the number of GB free of the total. Disks without this line are unformatted logical partitions that may be used for raw devices..
2. From the control panel, open **System and Security** > **Administrative Tools** > **Computer Management** > **Disk Management**.

   Avoid physical drives with existing logical partitions. Using the disk that contains the logical partitions for hard drives could damage the operating system. Using a disk partitioned into several raw logical partitions could damage those partitions. Use the raw device names available in the disk drive.

6.8 Viewing Access Permissions of a Raw Device

Display the current access permissions of a Windows raw device.

Procedure

1. Right-click **Command Prompt** and select **Run As Administrator**.
   "Administrator:" appears in the window title bar.
2. At the command prompt, type:

   ```
   rawaccess <filename>
   ```

   Where:
   - `<filename>` – the Windows universal namespace filename path to the raw device in one of the two formats:
     - `\\.\<logical_drive_name>`
     - `\\.\<physical_device_id>`
   - `<logical_drive_name>` – the partitioned letter used to represent the disk partition (for example: `\\.\D:`).
   - `<physical_device_id>` – the name assigned by the Window device manager. These names can be looked up using the Disk Manager in the control panel (for example: `\\.\PhysicalDisk1`).
   - `<permission_type>` – (optional) type of access action. If not specified, the current access list appears:
     - `DENY` – Adds an access denied entry to the specified device for the specified trustee.
     - `GRANT` – Adds an access allowed entry to the specified device for the specified trustees. The access allowed entry is merged with any existing inherited access allowed rights.
     - `REVOKE` – Removes an access allowed entry to the specified device for the specified trustee.
     - `SET` – Sets an access allowed entry to the specified device for the specified trustee.

     The permission_types comply with the Microsoft Access Control Entry (ACE) for Windows. See http://msdn.microsoft.com/en-us/library/windows/desktop/aa374899(v=vs.85).aspx

   Permissions display:

   ```
   1. Allow All   to COMPANY\user1 (User)
   2. Allow Exec  to \Everyone (Well Known Group)
   3. Allow All   to NT AUTHORITY\SYSTEM (Well Known Group)
   4. Allow All   to BUILTIN\Administrators (Alias)
   5. Allow Exec  to NT AUTHORITY\RESTRICTED (Well Known Group)
   ```

6.9 Setting Permissions for a Raw Device

You must set access permissions for a raw device on Windows before you can use it.

Context

By default, Windows does not allow direct write access to raw devices. To add permissions, use the `rawaccess` utility with the /GRANT option, which automatically merges the entry with default entries set up by the operating system.

Procedure

1. To use raw devices with SAP IQ, disable User Account Controls. In the control panel, choose your login from User Accounts and set **Change User Account Settings** to **Never Notify**.
2. Right-click **Command Prompt** and select **Run As Administrator**.
   "Administrator:" appears in the window title bar.
3. At the command prompt, type:

   ```
   rawaccess <filename> [{/<permission_type>} <trustee>]
   ```

   Where:

   filename

   Is the Windows universal namespace filename path to the raw device in one of the two formats:
   - `\\.\<logical_drive_name>` – the partitioned letter used to represent the disk partition (for example: `\\.\D:`).
   - `\\.\<physical_device_id>` – the name assigned by the Window device manager. These names can be looked up using the Disk Manager in the control panel (for example: `\\.\PhysicalDisk1`).

   `rawaccess` uses the standard Microsoft universal naming conventions for Win32 device namespaces. See http://msdn.microsoft.com/en-ca/library/windows/desktop/aa365247(v=vs.85).aspx.

   permission_type

   (Optional) The type of access action. If not specified, the current access list appears:
   - DENY – Adds an access denied entry to the specified device for the specified trustee.
   - GRANT – Adds an access allowed entry to the specified device for the specified trustees. The access allowed entry is merged with any existing inherited access allowed rights.
   - REVOKE – Removes an access allowed entry to the specified device for the specified trustee.
   - SET – Sets an access allowed entry to the specified device for the specified trustee.

   The permission type comply with the Microsoft Access Control Entry (ACE) for Windows. See http://msdn.microsoft.com/en-us/library/windows/desktop/aa374899(v=vs.85).aspx

   trustee Is a valid login ID for the user to be granted permissions

4. View the access permissions to verify that they are set correctly. Enter the `rawaccess` command followed by the path to the raw device, for example:

   For example:

   ```
   rawaccess \\.\F:
   ```

   Grant access permission to the raw device `E:` to `user1`.

   ```
   rawaccess \\.\E: /Grant user@domain
   ```

   The access permissions for the raw device display:

   ```
   1. Allow All   to DOMAIN\user (User)
   2. Allow Exec  to \Everyone (Well Known Group)
   3. Allow All   to NT AUTHORITY\SYSTEM (Well Known Group)
   4. Allow All   to BUILTIN\Administrators (Alias)
   5. Allow Exec  to NT AUTHORITY\RESTRICTED (Well Known Group)
   ```

6.10 Estimating Space and Dbspaces Required

To avoid difficulties when a database or a particular dbspace is full, estimate dbspace requirements before you create the database and the objects in it.

Context

You can run SAP IQ stored procedures to estimate how much space and how many dbspaces your databases will require.

Procedure

1. Run `sp_iqestspace` to estimate the amount of space you will need to create a database, based on the number of rows in the underlying database tables. For each table that you plan to create:

   ```
   sp_iqestspace <table_name>, <rows>[, <iqpagesize>]
   ```

   The amount of space needed by each table is returned as “RAW DATA index_size”.
2. Run `sp_iqestdbspaces` to determine how many dbspace files to create from the given space and what size they should be. Use the total index sizes calculated in step number 5 as the `<minsize>` and `<maxsize>` parameters:

   ```
   sp_iqestdbspaces (<dbsize> [,<iqpagesize>]
     [,<minsize>] [,<maxsize>] ...
   ```

Results

All these calculations are estimates. Results vary based on the columns and indexes you create for your database. See *SAP IQ SQL Reference* > *System Procedures* for syntax and usage notes for each procedure.

6.11 Dbspace Management Example

This section illustrates the dbspace management process from creating a new database and adding objects and data to the database, through relocating objects and dropping the empty dbspace. This example includes sample SQL code and the output of the related system stored procedures.

Creating the Database Objects

Create a small database, `dbspacedb`, using:

```
CREATE DATABASE 'D:\IQ\dbspacedb'DBA USER DBA DBA PASSWORD My@IQD@tabase
     IQ PATH 'D:\IQ\dbspacedb.iq'
     IQ SIZE 10
     IQ RESERVE 100
     TEMPORARY SIZE 10
     TEMPORARY RESERVE 10
     JCONNECT OFF;
```

Connect to the `dbspacedb` database:

```
CONNECT DATABASE dbspacedb
     user DBA identified by My@IQD@tabase;
```

Add two dbspaces to the `dbspacedb` database:

```
CREATE DBSPACE dbspacedb2
USING FILE dbspace2 'D:\IQ\dbspacedb.iq2'
SIZE 10 RESERVE 20 MB;CREATE DBSPACE dbspacedb3
USING FILE dbspace3 'D:\IQ\dbspacedb.iq3'
SIZE 10 RESERVE 40 MB;
```

Changing the Size of a Dbspace

The main store in the preceding example is too small for the tables to be added in the next example. The `ALTER DBSPACE` commands in this section change the dbspace size.

The database `dbspacedb` has a reserve size of 100 MB for the IQ main store, which was set using the `IQ RESERVE` parameter of the `CREATE DATABASE` statement. This IQ main store (the `IQ_SYSTEM_MAIN` dbspace) can be extended by 100 MB. The original `IQ_SYSTEM_MAIN` is created with a size of 10 MB (the `IQ SIZE`

parameter of `CREATE DATABASE`). The following `ALTER DBSPACE` command with the `ADD` parameter extends the `IQ_SYSTEM_MAIN` dbspace by 10 MB to 20 MB:

```
ALTER DBSPACE IQ_SYSTEM_MAIN ADD 10 mb;
sp_iqdbspace;
```

DBSpaceName	DBSpaceType	Writable	Online
IQ_SYSTEM_MAIN	MAIN	T	T
IQ_SYSTEM_TEMP	TEMPORARY	T	T
dbspacedb2	MAIN	T	T
dbspacedb3	MAIN	T	T

Usage	TotalSize	Reserve	NumFiles	NumRWFiles
25	20M	90M	1	1
7	10M	10M	1	1
1	10M	20M	1	1
1	10M	40M	1	1

Stripingon	StripeSize	BlkTypes	OK ToDrop
T	1K	1H,1248F,32D,177A,128M	N
T	1K	1H,64F,16A	N
T	1K	1H	Y
T	1K	1H	Y

If you do not create the `dbspacedb` database with an `IQ RESERVE` value, you cannot extend the dbspace. You can, however make the dbspace smaller, and the size taken away from the dbspace is added to the reserve.

The `IQ_SYSTEM_MAIN` dbspace is now 20 MB in size. Resize the dbspace to 15 MB:

```
ALTER DBSPACE IQ_SYSTEM_MAIN SIZE 15 mb;
sp_iqdbspace;
```

DBSpaceName	DBSpaceType	Writable	Online
IQ_SYSTEM_MAIN	MAIN	T	T
IQ_SYSTEM_TEMP	TEMPORARY	T	T
dbspacedb2	MAIN	T	T
dbspacedb3	MAIN	T	T

Usage	TotalSize	Reserve	NumFiles	NumRWFiles
25	15M	95M	1	1
7	10M	10M	1	1
1	10M	20M	1	1

Usage	TotalSize	Reserve	NumFiles	NumRWFiles
1	10M	40M	1	1

Stripingon	StripeSize	BlkTypes	OK ToDrop
T	1K	1H,1248F,32D,177A,128M	N
T	1K	1H,64F,16A	N
T	1K	1H	Y
T	1K	1H	Y

You can decrease the dbspace size only if the truncated portion is not in use. Use `sp_iqdbspaceinfo` to determine which blocks are in use by the objects on a dbspace.

Adding Database Objects

Create two tables in the `dbspacedb` database, create indexes, and add some data:

```
CREATE TABLE t1(c1 int);
CREATE TABLE t2(c1 int);
CREATE hg INDEX t1c1hg ON t1(c1);
CREATE hng INDEX t2c1hng ON t2(c1);
INSERT t1 VALUES(1);
INSERT t2 VALUES(2);
COMMIT;
```

Displaying Information about Dbspaces

Display information about all dbspaces in the `dbspacedb` database. The following example shows dbspaces in the iqdemo database. Output is divided into two parts to improve readability:

```
sp_iqdbspace;
```

DBSpaceName	DBSpaceType	Writable	Online	Usage	TotalSize	Reserve
IQ_SYSTEM_MAIN	MAIN	T	T	25	10M	100M
IQ_SYSTEM_TEMP	TEMPORARY	T	T	7	10M	10M
dbspacedb2	MAIN	T	T	1	10M	20M
dbspacedb3	MAIN	T	T	1	10M	40M

NumFiles	NumRWFiles	Stripingon	StripeSize	BlkTypes	OK ToDrop
1	1	T	1K	1H,1248F,32D,177A,128M	N
1	1	T	1K	1H,64F,16A	N

NumFiles	NumRWFiles	Stripingon	StripeSize	BlkTypes	OK ToDrop
1	1	T	1K	1H	Y
1	1	T	1K	1H	Y

Display information about object placement and space usage for a specific dbspace.

> **i** Note
>
> The next two examples show objects in the `iqdemo` database to better illustrate output. Note that `iqdemo` includes a sample user dbspace named `iq_main` that may not be present in your own databases.

The following output is divided into parts to improve readability:

```
sp_iqdbspaceinfo;
```

dbspace_name	object_type	owner	object_name	object_id	id
iq_main	table	DBA	emp1	3,813	743
iq_main	table	DBA	iq_dummy	3,801	742
iq_main	table	DBA	sale	3,822	744
iq_main	table	GROUPO	Contacts	3,662	734
iq_main	table	GROUPO	Customers	3,639	733
iq_main	table	GROUPO	Departments	3,756	740
iq_main	table	GROUPO	Employees	3,765	741
iq_main	table	GROUPO	FinancialCodes	3,736	738
iq_main	table	GROUPO	FinancialData	3,745	739
iq_main	table	GROUPO	Products	3,717	737
iq_main	table	GROUPO	SalesOrderItems	3,704	736
iq_main	table	GROUPO	SalesOrders	3,689	735

columns	indexes	metadata	primary_key
96K	0B	1.37M	0B
24K	0B	464K	0B
96K	0B	1.22M	0B
288K	0B	5.45M	24K
240K	48K	4.63M	24K
72K	0B	1.78M	24K
408K	0B	8.03M	24K
72K	0B	1.53M	24K
96K	0B	2.19M	24K
272K	192K	4.67M	24K

columns	indexes	metadata	primary_key
120K	0B	2.7M	24K
144K	0B	3.35M	24K

unique_ constraint	foreign_key	dbspace_online
0B	0B	Y
0B	0B	Y
0B	0B	Y
0B	48K	Y
0B	0B	Y
0B	48K	Y
0B	48K	Y
0B	0B	Y
0B	48K	Y
0B	0B	Y
0B	104K	Y
0B	144K	Y

Use the `sp_iqindexinfo` system stored procedure to display object placement and space usage for a specific table or index. The following information is from the `iqdemo` database:

```
sp_iqindexinfo 'table GROUPO.Customers';
```

Object	DBSpaceName	ObjSize	DBSpPct	MinBlk	MaxBlk
GROUPO.Customers	iq_main	200K	1	1,045,460	1,051,032
GROUPO.Customers.ASIQ_IDX_T733_C10_FP	iq_main	440K	1	1,046,689	1,047,147
GROUPO.Customers.ASIQ_IDX_T733_C1_FP	iq_main	440K	1	1,046,641	1,047,213
GROUPO.Customers.ASIQ_IDX_T733_C2_FP	iq_main	440K	1	1,046,961	1,047,203
GROUPO.Customers.ASIQ_IDX_T733_C3_FP	iq_main	440K	1	1,046,833	1,047,196
GROUPO.Customers.ASIQ_IDX_T733_C4_FP	iq_main	440K	1	1,046,737	1,047,189

Object	DBSpaceName	ObjSize	DBSpPct	MinBlk	MaxBlk
GROUPO.Customers.ASIQ_IDX_T733_C5_FP	iq_main	440K	1	1,046,929	1,047,182
GROUPO.Customers.ASIQ_IDX_T733_C6_FP	iq_main	440K	1	1,047,009	1,047,175
GROUPO.Customers.ASIQ_IDX_T733_C7_FP	iq_main	440K	1	1,046,945	1,047,168
GROUPO.Customers.ASIQ_IDX_T733_C8_FP	iq_main	440K	1	1,046,785	1,047,161
GROUPO.Customers.ASIQ_IDX_T733_C9_FP	iq_main	440K	1	1,046,881	1,047,154
GROUPO.Customers.ASIQ_IDX_T733_I11_HG	iq_main	152K	1	1,047,121	1,047,206
GROUPO.Customers.IX_customer_name	iq_main	304K	1	1,050,995	1,051,038

For the full syntax of the `sp_iqdbspace`, `sp_iqdbspaceinfo`, and `sp_iqindexinfo` system stored procedures, see *SAP IQ SQL Reference*.

6.12 Data Storage

A DBA can determine which tables and indexes reside on a given dbspace, relocate objects to other dbspaces, and drop any dbspace after emptying it of data.

A DBA can also define the number of writes to each dbspace before the disk striping algorithm moves to the next stripe.

In this section:

Dbfile Attributes and Operations [page 115]
A dbfile has read-write or read-only status.

Dbspace Attributes and Operations [page 118]
Dbspace statuses can be online, offline, or dynamically offline.

Read-Only and Read-Write Dbspaces and Files [page 121]
A file is read-only when either the file status is read-only or the file status is read-write, but the owning dbspace status is read-only.

6.12.1 Dbfile Attributes and Operations

A dbfile has read-write or read-only status.

A dbfile is read-write when it is added, and its runtime read-write status depends on both the read-write status of the dbspace and of the dbfile. The administrator can alter the read-write/read-only status of a dbfile, but cannot alter the online/offline status of a dbfile.

Operations that can be performed on dbfiles include adding, dropping, renaming logical name, and renaming the file path.

In this section:

Adding a File to the RLV Dbspace [page 115]
You can add a file to the RLV dbspace for extra capacity in storing RLV transaction logs.

Dropping a File from the RLV Dbspace [page 116]
You can remove a file from the RLV dbspace, provided that it is not the only file, and it is not in use.

Dropping Dbfiles from IQ_SYSTEM_TEMP [page 117]
Issue a `DROP DBSPACE` command to remove a dbfile.

Dbfiles and Output Files [page 117]
The `SYSDBFILE` view shows all the dbfiles in your database, including the catalog dbspace file, the IQ message file, dbfiles in the IQ main and temporary dbspaces, the transaction log file, and the SA temporary file.

6.12.1.1 Adding a File to the RLV Dbspace

You can add a file to the RLV dbspace for extra capacity in storing RLV transaction logs.

Prerequisites

- Requires the MANAGE ANY DBSPACE system privilege.
- For multiplex, you are connected to the coordinator.
- The RLV dbspace exists on the database and is online.
- If the dbspace is read-write, no RLV-enabled objects exist.

Context

Due to the nature of in-memory RLV storage, you cannot specify files as being READ-ONLY.

Procedure

To add a file to the RLV dbspace, execute the following:

```
ALTER DBSPACE <dbspace-name> ADD FILE <filename>
```

This statement added the file `rlv2` to the dbspace `d1`:

```
ALTER DBSPACE d1 ADD FILE 'rlv2.iq'
```

6.12.1.2 Dropping a File from the RLV Dbspace

You can remove a file from the RLV dbspace, provided that it is not the only file, and it is not in use.

Prerequisites

- Requires the MANAGE ANY DBSPACE system privilege.
- For multiplex, you are connected to the coordinator.
- The RLV dbspace is read-only.
- If the dbspace is read-write, the file is not in use.

Procedure

To drop the dbfile, execute:

```
ALTER DBSPACE <dbspace_name> DROP FILE <dbspace_filename>
```

This example drops the dbfile `rlv2` from the dbspace `d1`:

```
ALTER DBSPACE d1 DROP FILE rlv2
```

6.12.1.3 Dropping Dbfiles from IQ_SYSTEM_TEMP

Issue a `DROP DBSPACE` command to remove a dbfile.

Prerequisites

To drop a dbfile:

- Requires the MANAGE ANY DBSPACE system privilege.
- The dbfile cannot contain any data from user tables. SAP IQ does not allow you to drop a dbfile unless it is empty.
- There must be at least one read-write dbfile available in the dbspace.
- You must commit or roll back all transactions that use older versions of tables on the dbfile.

Procedure

1. Search the global temporary table schema using the system tables. Run DROP TABLE commands to relocate all objects resident on the dbfile:

   ```
   SELECT * FROM systable WHERE table_type = 'GBL TEMP'
   ```

2. Verify that all space on the dbfile is free:

   ```
   sp_iqfile IQ_SYSTEM_TEMP
   ```

   If the `OkToDrop` column contains "Y", you may drop the dbfile.

3. Drop the dbfile. For a multiplex database, use an absolute path:

   ```
   ALTER DBSPACE IQ_SYSTEM_TEMP DROP FILE FileHist3
   ```

6.12.1.4 Dbfiles and Output Files

The `SYSDBFILE` view shows all the dbfiles in your database, including the catalog dbspace file, the IQ message file, dbfiles in the IQ main and temporary dbspaces, the transaction log file, and the SA temporary file.

Files that are not dbfiles do not appear in the `SYSDBFILE` view. These include files that may be generated on server startup, such as the console log (specified by the `-o` switch) and the SQL log (specified by `-zo`). These log files do appear as database properties or server properties and may be examined by stored procedures such as `sa_db_properties( )` or the system function `db_property( )`.

6.12.2 Dbspace Attributes and Operations

Dbspace statuses can be online, offline, or dynamically offline.

Dynamically offline means that the dbspace is marked offline in memory, as opposed to marked offline in the catalog. If a database starts and one or more dbfiles cannot be opened, the database starts but the dbspace is marked dynamically offline. An administrator can use `ALTER DBSPACE ONLINE` to bring a dbspace back online after fixing a problem, but this changes only the dbspace's in-memory status.

> **i** Note
>
> Table data is inaccessible if any indexes, data, or partitions exist in an offline dbspace.

In addition to online, offline, or dynamically offline status, a dbspace also has read-write or read-only status. When created, a dbspace is online and read-write.

A dbspace also has striping attributes. An administrator may specify whether striping is on, and the stripe size.

You can create, alter, or drop dbspaces.

For multiplex dbspaces, see *iqmpx*. To change the status of a dbspace in SAP IQ Cockpit, see the SAP IQ Cockpit for SAP IQ online help in SAP IQ Cockpit or at https://help.sap.com/viewer/p/SAP_IQ.

In this section:

Dbspace Renaming Guidelines [page 118]
: You can rename a dbspace or dbfile name, but you cannot rename or drop catalog dbspace SYSTEM, IQ main dbspace IQ_SYSTEM_MAIN, IQ temporary dbspace IQ_SYSTEM_TEMP, shared temporary dbspace IQ_SHARED_TEMP, and IQ message dbspace IQ_SYSTEM_MSG.

Creating Dbspaces [page 119]
: Create a dbspace using the `CREATE DBSPACE` statement.

Guidelines for Dropping a Dbspace [page 120]
: Issue a `DROP DBSPACE` command to remove a database file.

6.12.2.1 Dbspace Renaming Guidelines

You can rename a dbspace or dbfile name, but you cannot rename or drop catalog dbspace SYSTEM, IQ main dbspace IQ_SYSTEM_MAIN, IQ temporary dbspace IQ_SYSTEM_TEMP, shared temporary dbspace IQ_SHARED_TEMP, and IQ message dbspace IQ_SYSTEM_MSG.

You can rename the logical name of files in IQ_SYSTEM_MAIN, IQ_SYSTEM_TEMP, and you can change the logical name of IQ_SYSTEM_MSG files, but you cannot change the logical name of files in SYSTEM. You cannot use ALTER DBSPACE RENAME TO to rename dbspaces IQ_SYSTEM_MAIN or IQ_SYSTEM_TEMP, IQ_SYSTEM_MSG, or SYSTEM.

6.12.2.2 Creating Dbspaces

Create a dbspace using the `CREATE DBSPACE` statement.

A new dbspace can be on the same or a different disk drive as the existing dbspaces. Requires the MANAGE ANY DBSPACE system privilege.

See *Maximum Limits* in SAP IQ SQL Reference for the maximum sizes of dbspaces on raw devices and operating system files. Some platforms require that you enable large file system files to reach the maximum size.

You can specify `SIZE` and `RESERVE` only for the IQ store and IQ temporary store, not for the catalog store.

Create main stores on raw devices.

When you specify a raw device for a new dbspace, SAP IQ automatically determines its file size and allocates the entire device for use as an IQ store. This may have unpredictable results on a file device.

If you indicate that the device is not raw, you can then specify the file size. The wizard verifies that the given path exists. If not, SAP IQ returns an error.

When you allocate file system files for dbspaces (System, IQ main, or IQ temporary), do not place the files on a file system that is shared over a local area network. Doing so can lead to poor I/O performance and other problems, including overloading the local area network. Do not place IQ dbspace files on network drives or (in a multiplex) Network File System (NFS) file systems. Simplex IQ servers support dbspace files on NFS.

To avoid conflicts, restrict the management of dbspaces to a single database administrator on a single connection.

Example

The following command creates a new dbspace called `library` that points to an operating system file named `library.iq` in the same directory as the `IQ_SYSTEM_MAIN` dbspace:

```
CREATE DBSPACE library
USING FILE library
'library.iq' SIZE 100 MB IQ STORE
```

To create a dbspace in SAP IQ Cockpit, see the SAP IQ Cockpit for SAP IQ online help in SAP IQ Cockpit or at https://help.sap.com/viewer/p/SAP_IQ.

After you add or drop a dbspace, issue a `CHECKPOINT`. System recovery begins after the most recent checkpoint.

How the Number of Dbspaces Affects Resource Use and Performance

The maximum number of dbspaces per database is an operating system limit that you can adjust; up to 2,047 dbspaces per database and 12 dbspaces for the catalog store. Increasing the number of dbspaces has no real impact on memory use or performance; avoid situations where you approach the maximum.

> **i Note**
>
> On HP and AIX platforms, overlapped I/O performance improves when you divide data among more dbspaces.

When data is stored on raw partitions, you can have one dbspace per drive. See *Maximum Limits* in *SAP IQ SQL Reference* for dbspace size limits.

When data is stored in a file system, you can take advantage of striping in the storage system. If you use operating system or hardware striping on a multiuser system, your stripe size should be at least 1 MB, or the highest size possible. The stripe size should be several times your IQ page size. You can also configure SAP IQ to perform software striping.

Before adding dbspaces, consider estimating your space requirements. For the most efficient resource use, make your dbspaces small enough to fit on your backup media, and large enough to fill up the disk.

6.12.2.3 Guidelines for Dropping a Dbspace

Issue a `DROP DBSPACE` command to remove a database file.

To drop a dbspace:

- Requires the MANAGE ANY DBSPACE system privilege.
- It cannot contain any data from user tables. SAP IQ does not allow you to drop a dbspace unless it is empty.
- It cannot be a required dbspace: `SYSTEM`, `IQ_SYSTEM_MAIN`, `IQ_SYSTEM_TEMP`, or `IQ_SYSTEM_MSG`. These dbspaces can never be dropped, but you may drop other dbspaces from the IQ main store or catalog store if the dbspace contains no user-created objects.

To empty a dbspace, you must:

- Relocate or drop all objects resident on the dbspace.
- Commit or roll back only transactions that are using older versions of tables.

Because of the way SAP IQ fills dbspaces with data, it is unlikely that a dbspace will become empty only after explicitly relocating tables, especially if disk striping is in use. Typically, you cannot empty a dbspace by truncating the tables in it, as even an empty table takes some space. To relocate the tables, use `ALTER TABLE MOVE`.

If you relocate a table while other users are reading from it, the normal rules of table versioning apply, that is, old table versions persist until the transactions of the readers complete.

A DBA can determine the dbspace in which tables and indexes are located by running the stored procedures `sp_iqspaceinfo`, `sp_iqdbspaceinfo`, and `sp_iqindexinfo`. These procedures show the number of blocks used by each table and index in each dbspace.

To find out whether you can drop a particular dbspace, run `sp_iqdbspace`. Look at the Block Types column (Blk Types), which tells you the contents of each dbspace. You can drop a dbspace if it contains block types “H,” “F,” “R,” “B,” “X,” and “C,” but not other block types.

Block type “A” is data from active table versions. Use `sp_iqdbspaceinfo` to determine which tables to relocate.

Block type “O” indicates old versions that may still be in use. You must roll back or commit active connections to release this space. Block type “M” indicates multiplex.

For instructions on using SAP IQ Cockpit to delete a dbspace, see the SAP IQ Cockpit for SAP IQ online help in SAP IQ Cockpit or at https://help.sap.com/viewer/p/SAP_IQ.

6.12.3 Read-Only and Read-Write Dbspaces and Files

A file is read-only when either the file status is read-only or the file status is read-write, but the owning dbspace status is read-only.

Altering a dbspace to read-only does not alter the catalog status of its associated files to read-only, but does make the associated files read-only at the operating system level. In other words, the file’s catalog read-only or read-write status remains the same, but data in the file cannot be modified.

For a read-only dbspace, the administrator can:

- Add a file
- Rename the file path of a dbfile in the dbspace (requires main dbspaces are offline)
- Drop an empty file
- Rename the dbspace or dbfile in the dbspace

In this section:

Status of Dbspaces and Associated Files [page 121]
A dbspace and its associated files can have individual read-only (RO) or read-write (RW) status, for example:

Allowed Dbspace Transformations [page 122]
Your ability to change a dbspace configuration depends on the type of alter operation and certain attributes of the dbspace. Alter operations are governed by the state of the dbspace (online or offline), the read-write status of the dbspace, and the type of dbspace.

6.12.3.1 Status of Dbspaces and Associated Files

A dbspace and its associated files can have individual read-only (RO) or read-write (RW) status, for example:

Object	Status	Effective Status	Dbspace of Table	Status of Table
dbspace1	RW	RW	dbspace1	RW
- file1	RO	RO		
- file2	RW	RW		
dbspace2	RO	RO	dbspace2	RO
- file1	RO	RO		
- file2	RW	RO		

Object	Status	Effective Status	Dbspace of Table	Status of Table
dbspace3	RW	RO	dbspace3	RO
- file1	RO	RO		
- file2	RO	RO		

A table is read-only when it is assigned to a read-only dbspace. A table partition is read-only when the partition is assigned to a read-only dbspace. No data modifications such as insert, delete, update, load, truncate table, and insert/delete/update through an updatable cursor are allowed to a read-only table or read-only table partition. No DDL operations such as ALTER TABLE add/drop column, create/drop index are allowed on a read-only table or read-only table partition.

Attempts to write to a read-only dbspace are detected when the modified pages are flushed to disk. Pages modified during an `INSERT...VALUES` statement are not written to the database until the next command that is not an `INSERT...VALUES` statement. (`INSERT...VALUES` is the only command that behaves this way.) SAP IQ returns an error for DDL operations on a read-only table or read-only table partition.

6.12.3.2 Allowed Dbspace Transformations

Your ability to change a dbspace configuration depends on the type of alter operation and certain attributes of the dbspace. Alter operations are governed by the state of the dbspace (online or offline), the read-write status of the dbspace, and the type of dbspace.

> **i** Note
>
> - Dynamically offline means the dbspace is marked offline in memory, as opposed to marked offline in the catalog.
> - A read-only (RO) IQ_SYSTEM_MAIN dbfile can be dynamically offline.
> - For IQ_SYSTEM_MSG, the only modification that is permitted is to rename the path, which is done using the command `ALTER DBSPACE IQ_SYSTEM_MSG RENAME 'filepath'.`

Online Dbspace

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER DBSPACE OFFLINE	Yes, if RO	No
ALTER DBSPACE ONLINE	No	Yes for IQ_SYSTEM_MAIN, No for IQ_SHARED_TEMP and IQ_SYSTEM_TEMP
ALTER DBSPACE READONLY	Yes, if it is RW	No
ALTER DBSPACE READWRITE	Yes, if it is RO	No

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER STRIPING or STRIPESIZEKB	Yes	Yes on simplex and on multiplex coordinator
RENAME DBSPACE	Yes	No
ADD FILE	Yes	Yes
DROP FILE	Yes, if empty and RO	Yes, if empty and RO. Dropping files in IQ_SHARED_TEMP requires that this operation be done only on the coordinator once it is started in a single-node mode. Dropping files in IQ_SHARED_TEMP in simplex is also supported. The first file that is made RW in IQ_SHARED_TEMP must be the last file to be dropped.
ALTER FILE READONLY	Yes, if RW	Yes, if RW and not the last RW dbfile. Making files in IQ_SHARED_TEMP read-only is not allowed.
ALTER FILE READWRITE	Yes, if RO	Yes, if RO
ALTER FILE SIZE	Yes, if RW dbspace and dbfile	Yes, if RW
ALTER FILE RENAME LOGICAL NAME	Yes	Yes
ALTER FILE RENAME PATH	No	Yes for IQ_SHARED_TEMP and IQ_SYSTEM_TEMP; takes effect when database restarts. No for IQ_SYSTEM_MAIN.

Offline Dbspace

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER DBSPACE OFFLINE	No	N/A
ALTER DBSPACE ONLINE	Yes	N/A
ALTER DBSPACE READONLY	No	N/A
ALTER DBSPACE READWRITE	No	N/A
ALTER STRIPING or STRIPESIZEKB	Yes	N/A
RENAME DBSPACE	Yes	N/A
ADD FILE	No	N/A
DROP FILE	Yes, if empty	N/A
ALTER FILE RO	Yes	N/A
ALTER FILE RW	Yes	N/A

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER FILE SIZE	No	N/A
ALTER FILE RENAME LOGICAL NAME	Yes	N/A
ALTER FILE RENAME PATH	Yes	N/A

Dynamically Offline Dbspace

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER DBSPACE OFFLINE	Yes, if RO	N/A
ALTER DBSPACE ONLINE	Yes	N/A for IQ_SYSTEM_MAIN and IQ_SYSTEM_TEMP, Yes for IQ_SHARED_TEMP
ALTER DBSPACE READONLY	Yes, if RW	N/A
ALTER DBSPACE READWRITE	No	N/A
ALTER STRIPING or STRIPESIZEKB	Yes	N/A
RENAME DBSPACE	Yes	N/A
ADD FILE	No	N/A
DROP FILE	Yes, if empty	N/A
ALTER FILE READONLY	No	N/A
ALTER FILE READWRITE	No	N/A
ALTER FILE SIZE	No	N/A
ALTER FILE RENAME LOGICAL NAME	Yes	N/A
ALTER FILE RENAME PATH	No	N/A

Read-Only Dbspace

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER DBSPACE OFFLINE	Yes, if online	N/A
ALTER DBSPACE ONLINE	Yes, if offline	N/A
ALTER DBSPACE READONLY	No	N/A
ALTER DBSPACE READWRITE	Yes, if online	N/A

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER STRIPING and STRIPESIZEKB	Yes	N/A
RENAME DBSPACE	Yes	N/A
ADD FILE	Yes	N/A
DROP FILE	Yes, if empty	N/A
ALTER FILE READONLY	Yes, if RW	N/A
ALTER FILE READWRITE	Yes, if RO	N/A
ALTER FILE SIZE	No	N/A
ALTER FILE RENAME LOGICAL NAME	Yes	N/A
ALTER FILE RENAME PATH	Yes, if offline	N/A

Read-Write Dbspace

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER DBSPACE OFFLINE	No	No
ALTER DBSPACE ONLINE	Yes, if dynamically offline	No for IQ_SYSTEM_MAIN and IQ_SYSTEM_TEMP, Yes if dynamically offline for IQ_SHARED_TEMP
ALTER DBSPACE READONLY	Yes	No
ALTER DBSPACE READWRITE	No	No
ALTER STRIPING and STRIPESIZEKB	Yes	Yes
RENAME DBSPACE	Yes	No
ADD FILE	Yes	Yes
DROP FILE	Yes, if empty	Yes, if empty and RO. Dropping files in IQ_SHARED_TEMP requires that this operation be done only on the coordinator once it is started in a single-node mode. Dropping files in IQ_SHARED_TEMP in simplex is also supported. The first file that is made RW in IQ_SHARED_TEMP must be the last file dropped.
ALTER FILE READONLY	Yes, if RW	Yes, if RW. Making files in IQ_SHARED_TEMP readonly is not allowed.
ALTER FILE READWRITE	Yes, if RO	Yes, if RO
ALTER FILE SIZE	Yes, if RW	Yes, if RW

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER FILE RENAME LOGICAL NAME	Yes	Yes
ALTER FILE RENAME PATH	No	Yes, takes effect when database restarts

Read-Only File

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER FILE READONLY	No	No
ALTER FILE READWRITE	Yes	Yes
ALTER FILE SIZE	No	No
ALTER FILE RENAME LOGICAL NAME	Yes	Yes
ALTER FILE RENAME PATH	Yes, if offline	Yes, takes effect when database restarts

Read-Write File

Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
ALTER FILE READONLY	Yes	Yes
ALTER FILE READWRITE	No	No
ALTER FILE SIZE	Yes, if dbspace is RW and ONLINE	Yes
ALTER FILE RENAME LOGICAL NAME	Yes	Yes
ALTER FILE RENAME PATH	No	Yes, takes effect when database restarts

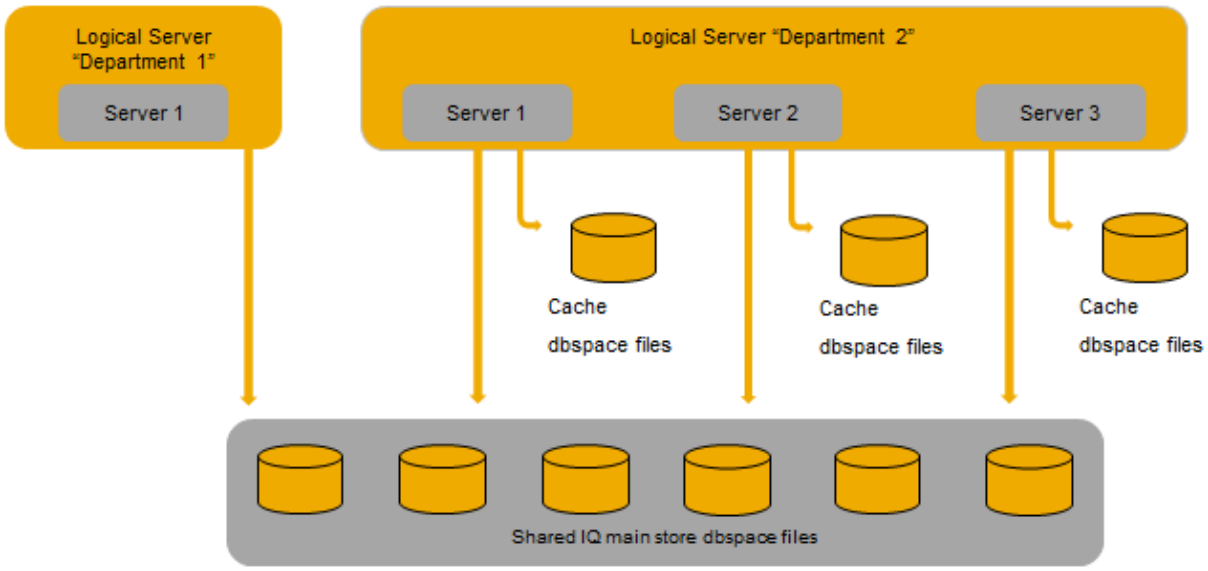
6.13 Cache Dbspace for Direct-Attached Storage

Add direct-attached storage devices, such as solid-state disks (SSDs), to a multiplex node or a simplex server by configuring a cache dbspace local to each node. Local direct-attached storage provides an SSD buffer pool extension to your database, which can improve I/O performance in general, and shared SAN device performance in particular. Although multiplex configurations gain the most benefit from direct-attached storage, simplex servers can benefit too.

The cache dbspace is an independent dbspace, like the IQ main store (IQ_SYSTEM_MAIN), IQ temporary store (IQ_SYSTEM_TEMP), and RLV (IQ RLV STOREs) dbspaces. Like all dbspaces, the cache dbspace can contain a single file or device, or multiple files or devices. You can configure a maximum of one cache dbspace per simplex and multiplex node.

> **i** Note
>
> Do not confuse the cache dbspace with the IQ main store main buffer cache. You set main buffer cache size using the database startup utility `start_iq` option `-iqmc` (main buffer cache size). See the *Utility Guide*.

After you create and configure the cache dbspace, the server caches copies of locally used read-only pages—both affinitized (preferred) and nonaffinitized pages—in the cache dbspace. In subsequent reads from the IQ main store dbspace files, the server first checks the cache dbspace to see if the page exists. If it does exist, the server reads the page from the high-speed SSD cache dbspace, rather than the shared SAN IQ main store dbspace, thereby improving performance.



You see maximum performance benefits if the cache dbspace is large enough to contain the node's entire working set of pages. If the cache dbspace is smaller than the working set of pages, the performance benefit is more modest.

The cache dbspace works with the data affinity feature to increase performance. Data affinity accesses cached data to boost query performance in multiplex servers. Affinity is automatically enabled for multiplex databases. Affinity tries to store the pages of partitions assigned to a node in the main buffer cache memory for that node. However, if an application's working set of pages exceeds the main buffer cache memory for the node, add direct-attached storage as a secondary fast access layer.

In this section:

Creating the Cache Dbspace [page 128]
Create a cache dbspace to add direct-attached storage (DAS) devices to a simplex or multiplex node. The cache dbspace increases performance by reducing reads from the storage area network. Each multiplex and simplex node can have one cache dbspace.

Dropping the Cache Dbspace [page 129]
Delete (drop) the cache dbspace from a simplex or multiplex node. Unlike other dbspaces, the cache dbspace does not need to be empty before you drop it.

Adding Storage to the Cache Dbspace [page 129]
Add additional direct-attached storage to the cache dbspace by specifying a file or device path.

Removing Storage from the Cache Dbspace [page 130]
Remove direct-attached storage from the cache dbspace by dropping files or raw devices.

Disabling and Reenabling the Cache Dbspace [page 131]
Change the online/offline status of a cache dbspace to disable a cache dbspace without dropping it, and to reenable a disabled cache dbspace.

Changing the Read-Only/Read-Write Status of a Cache Dbspace Dbfile [page 131]
Change a single dbfile in the cache dbspace to read-only or read-write status.

Monitoring the Cache Dbspace [page 132]
Use the `sp_iqstatus`, `sp_iqdbspace`, `sp_iqfile`, and `sp_iqcheckdb` stored procedures to monitor key performance indicators for cache dbspaces and their dbfiles.

6.13.1 Creating the Cache Dbspace

Create a cache dbspace to add direct-attached storage (DAS) devices to a simplex or multiplex node. The cache dbspace increases performance by reducing reads from the storage area network. Each multiplex and simplex node can have one cache dbspace.

Prerequisites

Requires MANAGE ANY DBSPACE system privilege.

Procedure

1. Connect to a simplex or multiplex node.
2. Create the cache dbspace and dbfiles. Specify the IQ CACHE STORE keywords in the CREATE DBSPACE statement.

```
CREATE DBSPACE myDAS
USING FILE iqdas1
'/dev/rdsk/ssd104' IQ CACHE STORE
```

6.13.2 Dropping the Cache Dbspace

Delete (drop) the cache dbspace from a simplex or multiplex node. Unlike other dbspaces, the cache dbspace does not need to be empty before you drop it.

Prerequisites

- A cache dbspace on the simplex or multiplex node.
- Requires MANAGE ANY DBSPACE system privilege.

Procedure

1. Connect to a simplex or multiplex node.
2. Drop the cache dbspace:

```
DROP DBSPACE myDAS
```

6.13.3 Adding Storage to the Cache Dbspace

Add additional direct-attached storage to the cache dbspace by specifying a file or device path.

Prerequisites

- A cache dbspace on the simplex or multiplex node.
- Requires MANAGE ANY DBSPACE system privilege.

Procedure

1. Connect to the simplex or multiplex node.
2. Specify a file or device path:

Option	Description
Specify a file	`ALTER DBSPACE myDAS` `ADD FILE iqdas2`

Option	Description
	`'sampledb.iqcache' SIZE 1024`

Specify a device path	`ALTER DBSPACE myDAS`
	`ADD FILE iqdas2`
	`'/dev/rdsk/clt2d3s4'`

6.13.4 Removing Storage from the Cache Dbspace

Remove direct-attached storage from the cache dbspace by dropping files or raw devices.

Prerequisites

- A cache dbspace on the simplex or multiplex node.
- Requires MANAGE ANY DBSPACE system privilege.

Context

You can drop dbfiles from a cache dbspace even if the dbspace is read-write.

Procedure

1. Connect to the simplex or multiplex node.
2. Remove a single file or raw device using the ALTER DBSPACE statement:

```
ALTER DBSPACE myDAS DROP FILE iqdas2
```

> **i** Note
>
> Multiple DROP FILE clauses are not allowed.

6.13.5 Disabling and Reenabling the Cache Dbspace

Change the online/offline status of a cache dbspace to disable a cache dbspace without dropping it, and to reenable a disabled cache dbspace.

Prerequisites

- A cache dbspace on the simplex or multiplex node.
- Requires MANAGE ANY DBSPACE system privilege.

Procedure

1. Connect to a simplex or multiplex node.
2. Disable the cache dbspace, or reenable a previously disabled cache dbspace, using the ALTER DBSPACE statement:

Option	Description
Disable a cache dbspace	`ALTER DBSPACE myDAS OFFLINE`
Re-enable a disabled cache dbspace	`ALTER DBSPACE myDAS ONLINE`

6.13.6 Changing the Read-Only/Read-Write Status of a Cache Dbspace Dbfile

Change a single dbfile in the cache dbspace to read-only or read-write status.

Prerequisites

- A cache dbspace on the simplex or multiplex node.
- Requires MANAGE ANY DBSPACE system privilege.

Procedure

1. Connect to a simplex or multiplex node.
2. Change a dbfile to read-only or read-write using the ALTER DBSPACE statement:

Option	Description
Change the cache dbspace dbfile to read-only	`ALTER DBSPACE myDAS ALTER FILE iqdas2 READONLY`
Change the cache dbspace dbfile to read-write	`ALTER DBSPACE myDAS ALTER FILE iqdas2 READWRITE`

6.13.7 Monitoring the Cache Dbspace

Use the `sp_iqstatus`, `sp_iqdbspace`, `sp_iqfile`, and `sp_iqcheckdb` stored procedures to monitor key performance indicators for cache dbspaces and their dbfiles.

Prerequisites

- `sp_iqstatus` requires the ALTER DATABASE, MANAGE ANY DBSPACE, MONITOR, or SERVER OPERATOR system privilege. Users without one of these system privileges must be granted EXECUTE privilege.
- `sp_iqdbspace` requires the MANAGE ANY DBSPACE system privilege. Users without MANAGE ANY DBSPACE system privilege must be granted EXECUTE privilege to run the stored procedure.
- `sp_iqfile` requires the MANAGE ANY DBSPACE system privilege. Users without the MANAGE ANY DBSPACE system privilege must be granted EXECUTE privilege.

Procedure

1. Run `sp_iqstatus`

```
...
 Main Store Out Of Space: N
  to view memory
              usage information about the cache dbspace and its dbfiles.
Number of Cache Dbspace Files: 5
 Number of Shared Temp DB Files: 0
 Shared Temp Store Out Of Space: N
 Number of Local Temp DB Files: 1
 Local Temp Store Out Of Space: N
 DB Blocks: 1-25600 IQ_SYSTEM_MAIN
 DB Blocks: 522208-547807 MainUser
```

```
Cache Dbspace Blocks: 1-5120 ssd_dev_1
Cache Dbspace Blocks: 522208-527327 ssd_dev_2
Cache Dbspace Blocks: 1044416-1049535 ssd_dev_3
Cache Dbspace Blocks: 1566624-1571743 ssd_dev_4
Cache Dbspace Blocks: 2088832-2093951 ssd_dev_5
Local Temp Blocks: 1-25600 IQ_SYSTEM_TEMP
Create Time: 2013-08-21 06:27:05.444
Update Time: 2013-08-21 06:27:14.035
Main IQ Buffers: 1588, 100Mb
Temporary IQ Buffers: 1588, 100Mb
Main IQ Blocks Used: 5250 of 38400, 13%=20Mb, Max Block#: 5313
Cache Dbspace IQ Blocks Used: 197 of 25600, 0%=0Mb, Max Block#: 0
Shared Temporary IQ Blocks Used: 0 of 0, 0%=0Mb, Max Block#: 0
Local Temporary IQ Blocks Used: 65 of 12800, 0%=0Mb, Max Block#: 0
Main Reserved Blocks Available: 12800 of 12800, 100%=50Mb
Shared Temporary Reserved Blocks Available: 0 of 0, 0%=0Mb
Local Temporary Reserved Blocks Available: 12800 of 12800, 100%=50Mb
...
```

Measurement	Description
Number of Cache Dbspace Files	The number of cache dbspace dbfiles in the database.
Cache Dbspace Block	Identifies the cache dbspace blocks and the corresponding storage device dbfile name.
Cache Dbspace IQ Blocks Used	The number of IQ blocks used, compared to the total number of IQ blocks. Usage is also shown as a percentage. If the percentage is high, consider adding more storage.

See *sp_iqstatus Procedure* to view memory in *SAP IQ SQL Reference* for detailed information.

2. Run `sp_iqdbspace` to view detailed information about the cache dbspace, including its read/write status and online/offline status. View the row for your cache dbspace name, as you specified in the CREATE DBSPACE statement (`myDAS` in this example). A cache dbspace displays `CACHE` in the `DBSpaceType` column:

```
IQ_SYSTEM_MAIN MAIN       T T 21 100M 0B 1 1 T 1K 1H,5088F,32D,128M N
IQ_SYSTEM_TEMP TEMPORARY  T T 1  100M 0B 1 1 T 1K 1H,64F,16A N
myDAS CACHE               T T 1  100M 0B 5 5 T 1K 5H,192F Y
MainUser MAIN             T T 1  100M 0B 1 1 T 1K 1H Y
```

Column	Description
`Writable`	T (writable) or F (not writable).
`Online`	T (online) or F (offline).
`Usage`	The percent of dbspace currently in use by all files in the dbspace.
`TotalSize`	The total size of all files in the dbspace in the units: ○ B (bytes) ○ K (kilobytes) ○ M (megabytes) ○ G (gigabytes) ○ T (terabytes) ○ P (petabytes)

Column	Description
Reserve	The total reserved space that can be added to all files in the dbspace.
NumFiles	The number of files in the dbspace.
NumRWFiles	The number of read-write files in the dbspace.
Stripingon	T (on) or F (off).
StripeSize	Always 1, if disk striping is on.
BlkTypes	The space used by both user data and internal system structures.
OkToDrop	"Y" indicates the dbspace can be dropped; otherwise "N".

3. Run `sp_iqfile` to display detailed information about each dbfile in the cache dbspace. View the rows for your cache dbspace name, as you specified in the `CREATE DBSPACE` statement (`myDAS` in this example):

```
IQ_SYSTEM_MAIN IQ_SYSTEM_MAIN exampleDB.iq MAIN RW T 21 100M 0B 1K 1H,5088F,
32D,128M 1 25600
MainUser MainUser exampleDB.iq2 MAIN RW T 1 100M 0B 1K 1H 522208 547807
myDAS ssd_dev_1 /dev/raw/ssd_dev_1 CACHE RW T 2 20M 0B 1K 1H,64F 1 5120
myDAS ssd_dev_2 /dev/raw/ssd_dev_2 CACHE RW T 1 20M 0B 1K 1H,32F 522208 527327
myDAS ssd_dev_3 /dev/raw/ssd_dev_3 CACHE RW T 1 20M 0B 1K 1H,32F 1044416
1049535
myDAS ssd_dev_4 /dev/raw/ssd_dev_4 CACHE RW T 1 20M 0B 1K 1H,32F 1566624
1571743
myDAS ssd_dev_5 /dev/raw/ssd_dev_5 CACHE RW T 1 20M 0B 1K 1H,32F 2088832
2093951
IQ_SYSTEM_TEMP IQ_SYSTEM_TEMP exampleDB.iqtmp TEMPORARY RW T 1 100M 0B 1K 1H,
64F,16A 1 25600
```

Column	Description
DBFileName	The logical file name.
Path	The location of the physical file or raw partition.
SegmentType	CACHE.
RWMode	The mode of the dbspace; always read-write (RW).
Online	○ T – online. This is the online value of both the file's associated dbspace and the file in SYS.ISYSIQDBFILE. ○ F – offline.
Usage	The percent of dbspace currently in use by this file in the dbspace. When run against a secondary node in a multiplex configuration, this column displays NA.
DBFileSize	The current size of the file or raw partition. For a raw partition, this size value can be less than the physical size.
Reserve	Reserved space that can be added to this file in the dbspace.
StripeSize	Always 1, if disk striping is on.

Column	Description
`BlkTypes`	The space used by both user data and internal system structures.
`FirstBlk`	The first IQ block number assigned to the file.
`LastBlk`	The last IQ block number assigned to the file.
`OkToDrop`	"Y" indicates the file can be dropped; otherwise "N".

See *sp_iqfile Procedure* in *SAP IQ SQL Reference* for detailed information.

Next Steps

You can use the `sp_iqcheckdb` stored procedure to perform database verifications against the cache dbspace. See *sp_iqcheckdb Procedure* in *SAP IQ SQL Reference*.

7 Manage SAP IQ Tables and Views

Create, alter, and delete tables and views in your SAP IQ databases.

Prerequisites

- With the SERVER OPERATOR system privilege, you can perform all data definition tasks. You also can grant this system privilege to other users to perform specific tasks. This includes the ability to grant the SERVER OPERATOR system privilege to other users.
- To create any database object, you need the CREATE ANY OBJECT system privilege for that type of object.
- Schedule data definition operations for times when database usage is low. All other users are blocked, though only briefly, from reading or writing to a table while you create or alter that table.
  For more information on concurrency rules during data definition, see *Table Locks for DDL Operations*.

> **i** Note
>
> SAP IQ consists of a catalog store and an IQ store. This section explains how you create objects in your IQ store. Tables created in the catalog store have the characteristics of SAP SQL Anywhere tables. To create tables in the catalog store, see the SAP SQL Anywhere documentation.

In this section:

Table Management [page 137]
: When you create a database, the only tables in it are the system tables, which hold the database schema.

Table Partitions [page 145]
: Table partitioning can improve performance by dividing large tables into smaller, more manageable storage objects. Partitions share the same logical attributes of the parent table, but can be placed in separate dbspaces and managed individually.

View Management [page 151]
: Views are computed tables.

Related Information

Table Locks for DDL Operations [page 220]

7.1 Table Management

When you create a database, the only tables in it are the system tables, which hold the database schema.

You may need to create, alter, and delete tables from a database. You can execute examples in documentation using `dbisql`, but the SQL statements are independent of the administration tool you are using.

You can create command files containing the `CREATE TABLE` and `ALTER TABLE` statements that define the tables in your database and store them in a source code control system. The command files allow you to re-create the database when necessary. They also let you create tables in a standardized way, which you can copy and revise.

In this section:

Creating a Table [page 137]
Create tables using Interactive SQL.

Altering a Table [page 142]
This section lists various examples that show how to change the structure of a table using the `ALTER TABLE` statement.

Dropping a Table [page 143]
Dropping a table deletes all the records in the table and then removes the table definition.

Creating Primary Keys [page 144]
Each row in a table is uniquely identified by its primary key.

Creating Foreign Keys [page 144]
Each foreign key relationship relates a candidate key (primary key and unique constraint) in one column to a column in another table, which becomes the foreign key.

Table Information in System Views [page 145]
Information about tables in a database is in the system view SYS.SYSTAB.

7.1.1 Creating a Table

Create tables using Interactive SQL.

Table Creation with Interactive SQL

The SQL statement for creating tables is `CREATE TABLE`.

This section describes how to use the `CREATE TABLE` statement. The examples in this section use the sample database. To try the examples, run `dbisql` and connect to the demo database with user ID `DBA`.

You can create tables with other tools in addition to Interactive SQL. The SQL statements described here are independent of the tool you are using.

Example

The following statement creates a new, permanent IQ table to describe qualifications of employees within a company. The table has columns to hold an identifying number, a name, and a type (say `technical` or `administrative`) for each skill:

```
CREATE TABLE skill (
    skill_id INTEGER NOT NULL,
    skill_name CHAR( 20 ) NOT NULL,
    skill_type CHAR( 20 ) NOT NULL
    )
```

You can execute this command by typing it into the `dbisql` command window, and pressing the execute key (F9).

- Each column has a data type. The `skill_id` is an `integer` (like 101), the `skill_name` is a fixed-width `CHARACTER` string containing up to 20 characters, and so on.
- The phrase NOT NULL after their data types indicates that all columns in this example must contain a value.
- In general, you would not create a table that has no primary key.

By internally executing the `COMMIT` statement before creating the table, SAP IQ makes permanent all previous changes to the database. There is also a `COMMIT` after the table is created.

> ⚠ Caution
>
> Altering or creating global or base tables can interfere with other users of the database. For large tables, `ALTER` or `CREATE TABLE` can be a time-consuming operation. `CREATE TABLE` processing delays execution of other IQ processes until the statement completes. Although you can execute `ALTER TABLE` statements while other connections are active, you cannot execute them while any other connection uses the table to be altered. `ALTER TABLE` processing excludes other requests referencing the table being offered while the statement processes.

Specifying Data Types

When you create a table, you specify the type of data that each column holds.

You can also define customized data types for your database. See *SAP IQ SQL Reference* for a list of supported data types. See the `CREATE DOMAIN` statement in *SAP IQ SQL Reference* for details on how to create a customized data type.

In this section:

Types of Tables [page 139]
SAP IQ recognizes four types of tables.

Automatic Index Creation for IQ Tables [page 141]
You can automate indexing for certain columns by creating a table with either `PRIMARY KEY` or `UNIQUE` constraints.

Optimizing Storage and Query Performance [page 141]

When you create a permanent table in an SAP IQ database, SAP IQ automatically stores permanent table columns in default indexes that facilitates a type of query called a projection.

7.1.1.1 Types of Tables

SAP IQ recognizes four types of tables.

- Base tables
- Local temporary tables
- Global temporary tables
- Join virtual tables

Base Tables are Permanent

Base tables are sometimes called main, persistent, or permanent tables because they are a permanent part of the database until you drop them explicitly.

They remain in the database over user disconnects, server restart, and recovery. Base tables and the data in them are accessible to all users who have the appropriate permissions. The `CREATE TABLE` statement shown in the previous example creates a base table.

Creating Temporary Tables

There are two types of temporary tables, global and local.

You **create** a global temporary table, using the `GLOBAL TEMPORARY` option of `CREATE TABLE`.

When you create a global temporary table, it exists in the database until it is explicitly removed by a `DROP TABLE` statement.

A database contains only one definition of a global temporary table, just as it does for a base table. However, each user has a separate instance of the data in a global temporary table. Those rows are visible only to the connection that inserts them. They are deleted when the connection ends, or commits. A given connection inherits the schema of a global temporary table as it exists when the user first refers to the table. Global temporary tables created on a multiplex server are also created on all other multiplex servers.

To select into a temporary table, use syntax like the following:

```
SELECT * INTO #TableTemp FROM lineitem
WHERE l_discount < 0.5
```

> **i** Note
>
> When writing scripts that `SELECT INTO` a temporary table, you should wrap any select list item that is not a base column in a `CAST` expression. This guarantees that the temporary table's column data type is the data type desired.

You **declare** a local temporary table for your connection only, using the `DECLARE LOCAL TEMPORARY TABLE` statement. A local temporary table exists until the connection ends or commits, or within a compound statement in which it is declared. The table and its data are completely inaccessible to other users.

An attempt to create a base table or a global temporary table will fail, if a local temporary table of the same name exists on that connection, as the new table cannot be uniquely identified by `<owner.table>`.

You can, however, create a local temporary table with the same name as an existing base table or global temporary table. References to the table name access the local temporary table, as local temporary tables are resolved first.

For example, consider the following sequence:

```
CREATE TABLE t1 (c1 INT);
INSERT t1 VALUES (9);
DECLARE LOCAL TEMPORARY TABLE t1 (c1 INT);
INSERT t1 VALUES (8);
SELECT * FROM t1;
```

The result returned is 8. Any reference to `t1` refers to the local temporary table `t1` until the local temporary table is dropped by the connection.

Dropping and Altering Global Temporary Tables

You drop a global temporary table just as you would a base table, with the `DROP TABLE` statement.

You cannot drop or alter a global temporary table while other connections are using the table.

In this section:

Placement of Tables [page 140]
SAP IQ creates tables in your current database.

7.1.1.1.1 Placement of Tables

SAP IQ creates tables in your current database.

If you are connected to an SAP IQ database, tables are placed as follows:

Type of Table	Permitted Placement
Permanent	• IQ store (default) • Catalog store
Global temporary	• IQ temporary store (default) • Catalog store

Type of Table	Permitted Placement
Local temporary	• IQ temporary store (default) • Catalog store Only visible to user who creates it

7.1.1.2 Automatic Index Creation for IQ Tables

You can automate indexing for certain columns by creating a table with either `PRIMARY KEY` or `UNIQUE` constraints.

These options cause SAP IQ to create an `HG` index for the column that enforces uniqueness.

If you use the `ALTER TABLE` command to add a `UNIQUE` column to an existing table, or to designate an existing column as `UNIQUE`, an `HG` index is created automatically.

7.1.1.3 Optimizing Storage and Query Performance

When you create a permanent table in an SAP IQ database, SAP IQ automatically stores permanent table columns in default indexes that facilitates a type of query called a projection.

SAP IQ optimizes this structure for query performance and storage requirements, based on these factors:

- The `IQ UNIQUE` option (`CREATE TABLE` or plug-in Column Properties page)
- The data type of the column and its width
- The `IQ PAGE SIZE` option (`CREATE DATABASE` or SAP IQ Cockpit Create Databases wizard)

In this section:

Effect of IQ UNIQUE [page 141]
`IQ UNIQUE` is an optional column constraint that explicitly defines the cardinality of a column and determines whether the column loads as `Flat FP` or `NBIT`.

7.1.1.3.1 Effect of IQ UNIQUE

`IQ UNIQUE` is an optional column constraint that explicitly defines the cardinality of a column and determines whether the column loads as `Flat FP` or `NBIT`.

`IQ UNIQUE` defines the expected cardinality of a column and determines whether the column loads as `Flat FP` or `NBit`. An `IQ UNIQUE(<n>)` value explicitly set to 0 loads the column as `Flat FP`. Columns without an IQ UNIQUE constraint implicitly load as `NBit` up to the limits defined by the `FP_NBIT_AUTOSIZE_LIMIT` and `FP_NBIT_LOOKUP_MB` options:

- FP_NBIT_AUTOSIZE_LIMIT limits the number of distinct values that load as NBit.
- FP_NBIT_LOOKUP_MB sets a threshold for the total NBit dictionary size.
- FP_NBIT_ROLLOVER_MAX_MB sets the dictionary size for implicit NBit rollovers from NBit to Flat FP.
- FP_NBIT_ENFORCE_LIMITS enforces NBit dictionary sizing limits. This option is OFF by default.

Using IQ UNIQUE with an <n> value less than the FP_NBIT_AUTOSIZE_LIMIT is not necessary. Auto-size functionality automatically sizes all low or medium cardinality columns as NBit. Use IQ UNIQUE in cases where you want to load the column as Flat FP or when you want to load a column as NBit when the number of distinct values exceeds the FP_NBIT_AUTOSIZE_LIMIT.

Indexes and IQ UNIQUE

If you estimate IQ UNIQUE incorrectly, there is no penalty for loads; the Optimizer simply uses the next larger index.

For queries, if you estimate IQ UNIQUE incorrectly and you have an HG, or storage-optimized default index, the Optimizer ignores the IQ UNIQUE value and uses the actual number of values in the index. If you do not have one of these indexes and your estimate is wrong by a significant amount (for example, if you specify IQ UNIQUE 1000000 when the actual number of unique values is 12 million), query performance may suffer.

To change the value of IQ UNIQUE for an existing index, run the sp_iqrebuildindex procedure.

Difference Between UNIQUE and IQ UNIQUE

IIQ UNIQUE(<n>) approximates the number of distinct values in a given column. An IQ UNIQUE(<n>) value explicitly set to '0' loads the column as Flat FP. A column without an IQ UNIQUE or a column with an IQ UNIQUE(<n>) value less than or equal to the limits defined by the FP_NBIT_AUTOSIZE_LIMIT and FP_NBIT_LOOKUP_MB options implicitly loads as NBIT. Each distinct value can appear many times. For example, in the employee table, a limited set of distinct values could appear in the state column, but each of those values could appear in many rows.

By contrast, when you specify UNIQUE or PRIMARY KEY, each value can occur only once in that column. For example, in the employee table, each value of ss_number, the employee's social security number, can occur just once throughout that column. This uniqueness extends to NULL values. Thus, a column specified as UNIQUE must also have the constraint NOT NULL.

7.1.2 Altering a Table

This section lists various examples that show how to change the structure of a table using the ALTER TABLE statement.

Adding a Column

This example adds a column to the `skill` table to allow space for an optional description of the skill:

```
ALTER TABLE skill
ADD skill_description CHAR( 254 )
```

Changing a Column Name

This example changes the name of the `skill_type` column to `classification`:

```
ALTER TABLE skill
RENAME skill_type TO classification
```

Deleting a Column

This example deletes the `classification` column:

```
ALTER TABLE skill
DELETE classification
```

Changing a Table Name

This example changes the name of the entire table:

```
ALTER TABLE skill
RENAME qualification
```

These examples show how to change the structure of the database. The `ALTER TABLE` statement can change many characteristics of a table—foreign keys can be added or deleted, and so on. However, you cannot use `MODIFY` to change table or column constraints. Instead, you must `DELETE` the old constraint and `ADD` the new one. In all these cases, once you make the change, stored procedures, views, and any other item referring to this column will no longer work.

7.1.3 Dropping a Table

Dropping a table deletes all the records in the table and then removes the table definition.

The following `DROP TABLE` statement deletes all the records in the `skill` table and then removes the definition of the `skill` table from the database:

```
DROP TABLE skill
```

Like the `CREATE` statement, the `DROP` statement automatically executes a `COMMIT` before and after dropping the table. This makes permanent all changes to the database since the last `COMMIT` or `ROLLBACK`.

The `DROP` statement also drops all indexes on the table.

If you only want to remove data rows but not the table itself, use the `TRUNCATE TABLE` statement. If you truncate a table while other users are reading from it, the normal rules of table versioning apply, that is, old table versions remain until the transactions of the readers complete.

`DROP TABLE` and `TRUNCATE TABLE` statements execute in seconds. The size of the data does not affect the speed of the operation.

7.1.4 Creating Primary Keys

Each row in a table is uniquely identified by its primary key.

The `CREATE TABLE` and `ALTER TABLE` statements allow many attributes of tables to be set, including column constraints and checks.

Creating a Primary Key

The following statement creates the same `skill` table as before, except that a primary key is added:

```
CREATE TABLE skill (
  skill_id INTEGER NOT NULL,
  skill_name CHAR( 20 ) NOT NULL,
  skill_type CHAR( 20 ) NOT NULL,
  primary key( skill_id )
)
```

The primary key values must be unique for each row in the table, which in this case, means that you cannot have more than one row with a given `skill_id`.

Columns in the primary key are not allowed to contain NULL. You must specify `NOT NULL` on the column in the primary key.

7.1.5 Creating Foreign Keys

Each foreign key relationship relates a candidate key (primary key and unique constraint) in one column to a column in another table, which becomes the foreign key.

Context

For example, you can create a table named `emp_skill`, which holds a description of each employee's skill level for each skill in which they are qualified, as follows:

```
CREATE TABLE emp_skill(
    emp_id INTEGER NOT NULL,
    skill_id INTEGER NOT NULL,
    "skill level" INTEGER NOT NULL,
    PRIMARY KEY( emp_id, skill_id ),
    FOREIGN KEY REFERENCES employee,
    FOREIGN KEY REFERENCES skill
    )
```

The `emp_skill` table definition has a primary key that consists of two columns: the `emp_id` column and the `skill_id` column. An employee may have more than one skill, and so appear in several rows, and several employees may possess a given skill, so that the `skill_id` may appear several times.

The `emp_skill` table also has two foreign keys. The foreign key entries indicate that the `emp_id` column must contain a valid employee number that is a primary key in the `employee` table from the `employee` table, and that the `skill_id` must contain a valid entry that is a primary key in the `skill` table from the `skill` table.

A table can only have one primary key defined, but it may have as many foreign keys as necessary.

You cannot create foreign key constraints on temporary tables of any kind—local, global, or automatic.

For more information about valid strings and identifiers, see *SAP IQ SQL Reference* > *SQL Language Elements*.

7.1.6 Table Information in System Views

Information about tables in a database is in the system view SYS.SYSTAB.

You can use the SAP IQ Cockpit Execute SQL window or Interactive SQL to browse the information in this view. Type the following statement in the command window to see all the columns in the SYS.SYSTAB view:

```
SELECT *
FROM SYS.SYSTAB
```

7.2 Table Partitions

Table partitioning can improve performance by dividing large tables into smaller, more manageable storage objects. Partitions share the same logical attributes of the parent table, but can be placed in separate dbspaces and managed individually.

> **i** Note
>
> Table data is inaccessible if partitioned data exists in an offline dbspace. Ensure all dbspaces are online.

In this section:

Restrictions on Table Partitions [page 146]
Some restrictions apply to table partitions.

Range Partitions [page 147]
Range partitioning divides large tables by a range of `<partition-key>` values established for each partition.

Hash Partitions [page 149]
Hash partitioning maps data to partitions based on `<partition-key>` values processed by an internal hashing function.

Hash-Range Partitions [page 150]
Hash-range partitioning is a composite partitioning scheme that subpartitions a hash-partitioned table by range.

Related Information

Manage Data Storage in SAP IQ [page 95]

7.2.1 Restrictions on Table Partitions

Some restrictions apply to table partitions.

- Range-partitions and composite partitioning schemes, like hash-range partitions, require the separately licensed VLDB Management option.
- Only base tables can be partitioned; global temporary tables or declared local temporary tables cannot. All rows of a table partition are physically collocated.
- Although range partitions or subpartitions can reside in separate dbspaces, individual dbspaces are recommended for `BLOB` or `CLOB` columns or `CHAR`, `VARCHAR` or `VARBINARY` columns greater than 255 bytes only.

Partition Keys

Partition keys cannot contain the following:

- `LOB` (`BLOB` or `CLOB`) columns
- `BINARY`, or `VARBINARY` columns
- `CHAR` or `VARCHAR` columns whose length is over 255 bytes
- `BIT` columns
- `FLOAT/DOUBLE/REAL` columns

Partition key columns can contain `NULL` and `DEFAULT` values. All `NULL` values are mapped to the same partition.

You can have up to 8 partitioning key columns for hash partitions and only one key column for range partitions or subpartitions.

You can define up to 1024 range partitions or subpartitions.

DML Operations

You can perform DML operations including `LOAD`, `INSERT`, `DELETE`, `TRUNCATE`, and `TRUNCATE TABLE PARTITION`. `UPDATE` of partition key columns is not supported and results in an error. You can perform the `UPDATE` of all other columns of a partitioned table.

SAP IQ generates an exception for DML operations on a `READ-ONLY` table or `READ-ONLY` table partition. `INSERT` and `LOAD` statements or `INSERT` by updatable cursor generate an error and operations roll back, if the given row does not fit into the specified range of partitions.

DDL Operations

Some restrictions apply to DROP, RENAME, PARTITION, UNPARTITION, MERGE, SPLIT, and MOVE partitions:

Operation	Restriction
Drop	You cannot drop a column from a partition key or range subpartition key. You cannot drop the last partition of a partitioned table.
Partition an unpartitioned table	An existing table cannot be made hash partitioned.
Merge two adjacent partitions	Both partitions must reside in the same dbspace. No data movement is required.
Split a partition	All rows must belong to the first of the two partitions after splitting. Split partition must be on same dbspace as original so that no data movement is required. You can split a range subpartition only if no data must be moved. All existing rows of the subpartition to be split must remain in the first subpartition after the split.
Move a partition to a new dbspace.	All rows of the partition are moved to data pages in the new dbspace. CREATE permission in the new dbspace is required.

Parent topic: Table Partitions [page 145]

Related Information

Range Partitions [page 147]
Hash Partitions [page 149]
Hash-Range Partitions [page 150]

7.2.2 Range Partitions

Range partitioning divides large tables by a range of <partition-key> values established for each partition.

As part of an information life cycle management strategy, range partitioning can shorten backup and restore times; provide a finer level of granularity for data validation; and support tiered storage.

In a <range-partitioning-scheme>, the <partition-key> is the column with value that determine the partition that the row belongs to:

```
<range-partitioning-scheme> ::=
    PARTITION BY RANGE( <partition-key> ) ( <range-partition-decl>[, <range-
partition-decl> … ] )
```

The <range-partition-declaration> determines how a named partition is placed in a dbspace:

```
<range-partition-declaration> ::=
```

```
    <range-partition-name> VALUES <= ( {<constant> |  MAX } ) [ IN <dbspace-
name> ]
```

The `VALUES` clause identifies the upper bound for each partition (in ascending order). Each range partition can be placed in its own dbspace and managed individually. Partition names must be unique within the set of partitions on a table.

Restrictions

Range partitioning is restricted to a single partition key column and a maximum of 1024 partitions.

Example

The most common form of range partitioning is to partition a table by date. This example creates a range partitioned table `<bar>` with six columns and three partitions, mapping data to partitions based on dates:

```
CREATE TABLE bar (
    c1 INT IQ UNIQUE(65500),
    c2 VARCHAR(20),
    c3 CLOB PARTITION (P1 IN Dsp11, P2 IN Dsp12,
        P3 IN Dsp13),
`   c4 DATE,
    c5 BIGINT,
    c6 VARCHAR(500) PARTITION (P1 IN Dsp21,
        P2 IN Dsp22),
    PRIMARY KEY (c5) IN Dsp2) IN Dsp1
    PARTITION BY RANGE (c4)
    (P1 VALUES <= ('2006/03/31') IN Dsp31,
     P2 VALUES <= ('2006/06/30') IN Dsp32,
     P3 VALUES <= ('2006/09/30') IN Dsp33);
```

Parent topic: Table Partitions [page 145]

Related Information

Restrictions on Table Partitions [page 146]
Hash Partitions [page 149]
Hash-Range Partitions [page 150]

7.2.3 Hash Partitions

Hash partitioning maps data to partitions based on <partition-key> values processed by an internal hashing function.

Hash partitioning distributes data to logical partitions for parallel execution, which can enhance join performance on large tables and distributed queries (DQP).

In a <hash-partitioning-scheme> declaration, a <partition-key> is a column or group of columns, whose composite value determines the partition where each row of data is stored:

```
<hash-partitioning-scheme> ::=
    PARTITION BY HASH  ( <partition-key>[ , <partition-key>, … ] )
```

Hash partition keys are restricted to a maximum of eight columns with a combined declared column width of 5300 bytes or less. For hash partitions, the table creator determines only the partition key columns; the number and location of the partitions are determined internally.

Restrictions

- You cannot add, drop, merge, or split a hash partition.
- You cannot add or drop a column from a hash partition key.

Examples

In this example, table `tbl42` includes a `PRIMARY KEY` (column c1) and a `HASH PARTITION KEY` (columns c4 and c3).

```
CREATE TABLE tbl42 (
  c1 BIGINT NOT NULL,
  c2 CHAR(2) IQ UNIQUE(50),
  c3 DATE IQ UNIQUE(36524),
  c4 VARCHAR(200),
  PRIMARY KEY (c1))
  PARTITION BY HASH ( c4, c3 )
```

This example shows the common case where the join key is both the primary key and the hash partitioning key.

```
CREATE TABLE bar (
  c1 BIGINT NOT NULL,
  c2 CHAR(2) IQ UNIQUE(50),
  c3 DATE IQ UNIQUE(36524),
  c4 VARCHAR(200),
  c5 CLOB,
  PRIMARY KEY ( c1 ) )
  PARTITION BY HASH ( c1 )
```

Parent topic: Table Partitions [page 145]

Related Information

Restrictions on Table Partitions [page 146]
Range Partitions [page 147]
Hash-Range Partitions [page 150]

7.2.4 Hash-Range Partitions

Hash-range partitioning is a composite partitioning scheme that subpartitions a hash-partitioned table by range.

Hash-range partitioning provides the benefits of hash partitioning and range partitioning. Hash partitioning provides the best distributed query performance; range sub-partitioning enhances administrative tasks as part of an information life cycle management strategy.

In a `<hash-range-partitioning-scheme>` declaration, a `SUBPARTITION BY RANGE` clause adds a new range subpartition to an existing hash-range partitioned table:

```
<hash-range-partitioning-scheme> ::=
    PARTITION BY HASH  ( <partition-key>[ , <partition-key>, … ] )
    [ SUBPARTITION BY RANGE
    ( <range-partition-decl>[ , <range-partition-decl> … ] ) ]
```

The hash partition specifies how the data is logically distributed and collocated; the range subpartition specifies how the data is physically placed. The new range subpartition is logically partitioned by hash with the same hash partition keys as the existing hash-range partitioned table. The range subpartition key is restricted to one column.

Examples

This example (table `tbl42`) includes a `PRIMARY KEY` (column c1) and a hash partition key (columns c4 and c2) and a range subpartition key (column c3) :

```
CREATE TABLE tbl42 (
    c1 BIGINT NOT NULL,
    c2 CHAR(2) IQ UNIQUE(50),
    c3 DATE,
    c4 VARCHAR(200),
  PRIMARY KEY (c1)) IN Dsp1

  PARTITION BY HASH ( c4, c2 )
  SUBPARTITION BY RANGE ( c3 )
  ( P1 VALUES <= ('2018/03/31') IN Dsp31,
    P2 VALUES <= ('2018/06/30') IN Dsp32,
    P3 VALUES <= ('2018/09/30') IN Dsp33) ;
```

In this example, table `tbl43` includes different dbspaces for storage of `RANGE` subpartitions for column c4. This syntax is similar to existing syntax for `RANGE` partitions. Note the use of `SUBPARTITION` in column c4:

```
CREATE TABLE tbl43 (
```

```
    c1 bigint not null,
    c2 char(2) iq unique(50),
    c3 date,
    c4 varchar(200),
    SUBPARTITION (P1 in Dsp331, P2 in Dsp332, P3 in Dsp333),
    PRIMARY KEY (c1)) IN Dsp1
    PARTITION BY HASH (c4, c2)
    SUBPARTITION BY RANGE (c3)
    (P1 VALUES <= ('2018/03/31') IN Dsp31,
     P2 VALUES <= ('2018/06/30') IN Dsp32,
     P3 VALUES <= ('2018/09/30') IN Dsp33) ;
```

Parent topic: Table Partitions [page 145]

Related Information

Restrictions on Table Partitions [page 146]
Range Partitions [page 147]
Hash Partitions [page 149]

7.3 View Management

Views are computed tables.

You can use views to show database users exactly the information you want to present, in a format you can control.

Similarities between Views and Base Tables

Views are similar to the permanent tables of the database (a permanent table is also called a base table) in many ways:

- You can assign access permissions to views just as to base tables.
- You can perform `SELECT` queries on views.
- You can perform `INSERT` and `DELETE` operations on some views.
- You can create views based on other views.

Differences Between Views and Permanent Tables

There are some differences between views and permanent tables:

- You cannot create indexes on views.

- `INSERT`, `DELETE`, and `UPDATE` operations can only be performed on certain views.
- You cannot assign integrity constraints and keys to views.
- Views refer to the information in base tables, but do not hold copies of that information. Views are recomputed each time you invoke them.

Benefits of Tailoring Data Access

Views are used to tailor access to data in the database. Tailoring access serves several purposes:

- By not allowing access to information that is not relevant.
- By presenting users and application developers with data in a more easily understood form than in the base tables.
- By centralizing in the database the definition of common queries.

In this section:

Creating Views [page 152]
A `SELECT` statement operates on one or more tables and produces a result set that is also a table.

Guidelines for Using Views [page 153]
There are certain restrictions, both on the `SELECT` statements you use to create views, and on your ability to insert into, delete from, or update them.

Guidelines for Modifying Views [page 154]
You can modify a view using the `ALTER VIEW` statement.

Permissions on Views [page 155]
A user may perform an operation through a view if one or more of the following are true:

How to Delete Views [page 155]
To delete a view from the database in Interactive SQL, use the `DROP` statement.

View Information in System Views [page 155]
Information about views in a database is in the system view SYS.SYSVIEW.

7.3.1 Creating Views

A `SELECT` statement operates on one or more tables and produces a result set that is also a table.

Like a base table, a result set from a `SELECT` query has columns and rows. A view gives a name to a particular query, and holds the definition in the database system tables.

Example

Suppose that you frequently need to list the number of employees in each department. You can get this list with the following statement:

```
SELECT DepartmentID, COUNT(*)
   FROM Employees
   GROUP BY DepartmentID
```

You can create a view containing the results of this statement as follows:

```
CREATE VIEW DepartmentSize AS
   SELECT DepartmentID, COUNT(*)
   FROM Employees
   GROUP BY DepartmentID
```

The information in a view is not stored separately in the database. Each time you refer to the view, the associated `SELECT` statement is executed to retrieve the appropriate data.

On one hand, this is good because it means that if someone modifies the `Employees` table, the information in the `DepartmentSize` view will be automatically up to date. On the other hand, complicated `SELECT` statements may increase the amount of time SQL requires to find the correct information every time you use the view.

7.3.2 Guidelines for Using Views

There are certain restrictions, both on the `SELECT` statements you use to create views, and on your ability to insert into, delete from, or update them.

Restrictions on SELECT Statements

You cannot use an ORDER BY clause in the `SELECT` query. A characteristic of relational tables is that there is no significance to the ordering of the rows or columns, and using an ORDER BY clause imposes an order on the rows of the view. You can use the GROUP BY clause, subqueries, and joins in view definitions.

Scalar value subqueries are supported only within the top-level `SELECT` list (not in a view, derived table, or subquery). Sometimes views or derived tables used in the FROM clause of the top-level `SELECT` are simple enough that they can be “flattened” up into the top-level `SELECT`. As a result of this, the preceding rule is actually enforced only for subqueries, nonflattened views, and nonflattened derived tables. For example:

```
CREATE VIEW test_view AS SELECT testkey,(SELECT COUNT(*) FROM tagtests WHERE
tagtests.testkey = testtrd.testkey ) FROM testtrd
```

```
SELECT * FROM test_view
Msg 21, Level 14, State 0:
SQL Anywhere Error -1005004: Subqueries are allowed only as arguments of
comparisons, IN, and EXISTS,
-- (opt_Select.cxx 2101)
```

To develop a view, tune the SELECT query by itself until it provides exactly the results you need in the format you want. Once you have the correct SELECT query, you can add a phrase in front of the query to create the view. For example:

```
CREATE VIEW <viewname> AS
```

Guidelines for Inserting and Deleting from Views

UPDATE, INSERT, and DELETE statements are allowed on some views, but not on others, depending on their associated SELECT statement.

You cannot update, insert into, or delete from views that contain:

- Aggregate functions, such as COUNT(*)
- A GROUP BY clause in the SELECT statement
- A UNION operation

In all these cases, there is no way to translate the UPDATE, INSERT, or DELETE into an action on the underlying tables.

> ⚠ Caution
>
> Do not delete views owned by the dbo user ID, which owns system objects. Deleting such views or changing them into tables may cause unexpected problems.

7.3.3 Guidelines for Modifying Views

You can modify a view using the ALTER VIEW statement.

The ALTER VIEW statement replaces a view definition with a new definition; it does not modify an existing view definition.

The ALTER VIEW statement maintains the permissions on the view.

Example

For example, to replace the column names with more informative names in the DepartmentSize view described above, you could use the following statement:

```
ALTER VIEW DepartmentSize
  (DepartmentID, NumEmployees)
AS
  SELECT DepartmentID, COUNT(*)
  FROM Employees
  GROUP BY DepartmentID
```

7.3.4 Permissions on Views

A user may perform an operation through a view if one or more of the following are true:

- The appropriate permissions on the view for the operation have been granted to the user by a DBA.
- The user has the appropriate permissions on all the objects (such as base tables, views, procedures) for the operation.
- The user was granted appropriate permissions for the operation on the view by a non-DBA user. This user must be either the owner of the view or have WITH GRANT OPTION of the appropriate permission(s) on the view. The owner of the view must be one of the following:
  - A DBA.
  - A non-DBA, but also the owner of all the objects referred to by the view.
  - A non-DBA, and not the owner of some or all of the objects referred to by the view, but the view owner has SELECT permission WITH GRANT OPTION on the objects not owned and any other required permission(s) WITH GRANT OPTION on the objects not owned for the operation.
    Instead of the owner having permission(s) WITH GRANT OPTION on the objects, permission(s) may have been granted to PUBLIC. This includes SELECT permission on system tables.

UPDATE permissions can be granted only on an entire view. Unlike tables, UPDATE permissions cannot be granted on individual columns within a view.

7.3.5 How to Delete Views

To delete a view from the database in Interactive SQL, use the `DROP` statement.

The following statement removes the DepartmentSize view:

```
DROP VIEW DepartmentSize
```

7.3.6 View Information in System Views

Information about views in a database is in the system view SYS.SYSVIEW.

You can use Interactive SQL to browse the information in this view. Type the following statement in the `dbisql` command window to see all the columns in the SYS.SYSVIEW view:

```
SELECT *
FROM SYS.SYSVIEW
```

To extract a text file containing the definition of a specific view, use a statement such as the following:

```
SELECT view_def FROM SYS.SYSVIEW
WHERE view_object_id = 1583;
OUTPUT TO viewtext.sql
FORMAT ASCII
```

See *SAP IQ SQL Reference > System Tables and Views*.

8 Index SAP IQ Columns

SAP IQ indexes are designed to make the best use of memory, disk and CPU cycles for specific types of queries.

- Index sizes usually remain small. The entire database can be fully indexed and made available for ad hoc queries in the same space that would be needed to store the raw data. Traditional databases often need three or more times more space.
- Queries are resolved by efficiently combining and manipulating indexes on only the relevant columns. This avoids time-consuming table scans.
- I/O is minimized, eliminating potential bottlenecks.
- Because indexes are compact, more data can be kept in memory for subsequent queries, thereby speeding throughput on iterative analysis.
- Tuning is data dependent, allowing data to be optimized once for any number of ad hoc queries.

In this section:

Overview of Indexes [page 157]
Indexes improve data retrieval performance. SAP IQ indexes differ from traditional indexes and are designed to accelerate queries in the data warehousing environment.

Index Types Comparison [page 157]
Choose the column index type appropriate to your task.

Criteria for Choosing Indexes [page 172]
The set of indexes that you define for any given column can have dramatic impact on the speed of query processing.

Creating Indexes [page 177]
SAP IQ automatically creates a default index on every table column when you create a database in an IQ store. For best query performance, create additional indexes.

Running the Index Advisor [page 179]
If you set the INDEX_ADVISOR option on your database, SAP IQ issues messages in the message log or query plan to suggest additional indexes that might improve performance.

Renaming Indexes [page 180]
You can rename an index in a base table or global temporary table with the owner type USER.

Viewing Indexes [page 180]
To see information about indexes, use the `sp_iqindex` stored procedure.

Removing Indexes [page 181]
Remove column indexes that are no longer required.

Retaining Indexes When Removing Foreign Key Constraints [page 182]
You may want to remove a foreign key constraint, but retain the underlying `HG` index. A non-unique `HG` index can provide query performance improvement, but may be expensive to build.

Optimizing Performance for Joins [page 182]
You can create indexes to optimize join performance.

8.1 Overview of Indexes

Indexes improve data retrieval performance. SAP IQ indexes differ from traditional indexes and are designed to accelerate queries in the data warehousing environment.

When you load data into a table, SAP IQ physically stores data by column rather than by row, for each column in the table. The column orientation gives IQ indexes important advantages over traditional row-based indexing.

The default column storage structure that SAP IQ creates for each column is an index that is optimized for storing and projecting data. The column indexes you define are created as part of each individual table.

Logically, you can still access the data as in traditional row-based SQL databases. Column storage structures your data according to the attributes you want to track. In a data warehousing environment, you typically look at specific attributes of thousands or millions of data rows, rather than complete, single rows of data. Column storage optimizes your ability to perform selections or calculations on specific attributes.

8.2 Index Types Comparison

Choose the column index type appropriate to your task.

Index Type	Purpose
Compare (CMP) Index [page 160]	Indexes relationship between any two distinct columns with identical data types, precision, and scale.
Containment (WD) Index [page 158]	Stores words from a column string of `CHAR`, `VARCHAR`, or `LONG VARCHAR` data.
Date (DATE) and Datetime (DTTM) Indexes [page 161]	Processes queries involving date or datetime quantities.
Fast Projection (FP) Index [page 166]	Optimizes projections and enables certain kinds of search conditions to be evaluated. Assigned by default to any permanent table.
High_Group (HG) Index [page 167]	Used for join columns with integer data types. Handles `GROUP BY` and equality operations efficiently. Recommended for columns with more than 1000 distinct values.
High_Non_Group (HNG) Index [page 169]	**i Note** HG offers better performance than HNG. We encourage you to use the HG index instead. Used for high-cardinality operations involving ranges or aggregates.
Low_Fast (LF) Index	**i Note** An HG index offers better performance than an LF index. Therefore, use HG indexes instead.

Index Type	Purpose
TEXT Index [page 169]	Used for unstructured data analytics functionality, an SAP IQ licensed option.
Time (TIME) Index [page 170]	Processes queries involving time quantities.

In this section:

Containment (WD) Index [page 158]
The Containment (`WD`) index allows you to store words from a column string of `CHAR`, `VARCHAR`, and `LONG VARCHAR` data.

Compare (CMP) Index [page 160]
A Compare (CMP) index is the relationship between two columns. You may create Compare indexes on any two distinct columns with identical data types, precision, and scale. The `CMP` index stores the binary comparison (<, >, or =) of its two columns.

Date (DATE) and Datetime (DTTM) Indexes [page 161]
`DATE` and `DTTM` index types process queries involving date or datetime quantities.

Fast Projection (FP) Index [page 166]
By default, the database engine automatically creates a Fast Projection (`FP`) index on all columns. `FP` indexes optimize projections and allow the database engine to evaluate certain kinds of search conditions.

High_Group (HG) Index [page 167]
The High_Group index is commonly used for join columns with integer data types. It is also more commonly used than High_Non_Group because it handles `GROUP BY` efficiently.

High_Non_Group (HNG) Index [page 169]
The `HNG` index was common for range searches. The `HG` index offers better performance.

TEXT Index [page 169]
To use `TEXT` indexes, you must be specifically licensed to use the unstructured data analytics functionality.

Time (TIME) Index [page 170]
Use the TIME index to process queries involving time quantities.

8.2.1 Containment (WD) Index

The Containment (`WD`) index allows you to store words from a column string of `CHAR`, `VARCHAR`, and `LONG VARCHAR` data.

To create `LONG VARCHAR` columns, you must be specifically licensed to use the unstructured data analytics functionality. See *SAP IQ Administration: User-Defined Functions*.

In this section:

Recommended Use of WD [page 159]
Use a `WD` index for the fastest access to columns that contain a list of keywords (for example, in a bibliographic record or Web page).

Advantages and Disadvantages of WD [page 160]
Advantages and disadvantages of using a WD index.

8.2.1.1 Recommended Use of WD

Use a WD index for the fastest access to columns that contain a list of keywords (for example, in a bibliographic record or Web page).

These restrictions apply to WD indexes:

- You cannot specify the UNIQUE attribute.
- It can be used only with the CONTAINS or LIKE predicate.
- The column name must identify a CHAR, VARCHAR, or LONG VARCHAR column in a base table.
- The minimum permitted column width is 3 bytes and the maximum permitted column width is the maximum width for a LOB column (the maximum length is equal to 4 GB multiplied by the database page size).
- You must enclose the list of delimiters in single quotes.
- You omit the DELIMITED BY clause or specify the <separators-string> value as empty (single quotes), then SAP IQ uses the default set of separators. The default set of characters includes all 7-bit ASCII characters that are not 7-bit ASCII alphanumeric characters, except the hyphen and the single quotation mark, which, by default, are part of words. There are 64 separators in the default separator set.
- If you specify multiple DELIMITED BY and LIMIT clauses, no error is returned, but only the last clause of each type is used. For example, the following two statements return identical results:
  - Statement 1:

    ```
    CREATE WD INDEX c1wd on foo(c1)
    DELIMITED BY 'f' LIMIT 40 LIMIT 99 DELIMITED BY 'g' DELIMITED BY 'h';
    ```

  - Statement 2:

    ```
    CREATE WD INDEX c1wd on foo(c1)
    DELIMITED BY 'h' LIMIT 99;
    ```

- After a WD index is created, any insertions into its column are parsed using the separators and maximum word size cannot be changed after the index is created.
  For CHAR columns, Specify a space as at least one of the separators or use the default separator set. SAP IQ automatically pads CHAR columns to the maximum column width. If your column contains blanks in addition to the character data, queries on WD indexed data may return misleading results. For example, column company_name contains two words delimited by a separator, but the second word is blank padded:

  ```
  'Concord' 'Farms                                          '
  ```

  Suppose that a user entered the following query:

  ```
  SELECT COUNT(*)FROM Customers WHERE CompanyName contains ('Farms')
  ```

  The parser determines that the string contains the following, instead of 'Farms':

  ```
  'Farms                                                '
  ```

  The parser then returns 1. You can avoid this problem by using VARCHAR instead of CHAR columns.

- The `sp_iqcheckdb` (DBCC consistency checker) allocation, check, verify, and repair modes support the `WD` index on `CHAR`, `VARCHAR`, and `LONG VARCHAR` columns.

8.2.1.2 Advantages and Disadvantages of WD

Advantages and disadvantages of using a `WD` index.

Advantages	Disadvantages
• Huge performance gains are possible for large loads. • Certain LIKE predicates execute faster with this index. • CONTAINS predicate used with this index takes precedence over the LIKE predicate. • Best way to index keywords or parts of a URL.	• Disk space requirements may potentially be very large. • Index not possible if uniqueness enforced. • Can use this index only if data in your columns is `CHAR`, `VARCHAR`, or `LONG VARCHAR`.

8.2.2 Compare (CMP) Index

A Compare (CMP) index is the relationship between two columns. You may create Compare indexes on any two distinct columns with identical data types, precision, and scale. The `CMP` index stores the binary comparison (<, >, or =) of its two columns.

In this section:

Recommended Use of CMP [page 160]
You can create a `CMP` index on columns that are NULL, NOT NULL, or a mixture. The `CMP` index cannot be unique.

Restrictions on CMP Indexes [page 161]
The following restrictions apply to `CMP`.

8.2.2.1 Recommended Use of CMP

You can create a `CMP` index on columns that are NULL, NOT NULL, or a mixture. The `CMP` index cannot be unique.

Numeric and decimal data types are considered identical. You may create `CMP` indexes on them when precision and scale are identical. For `CHAR`, `VARCHAR`, `BINARY`, and `VARBINARY` columns, precision means having the same column width.

Example

For example, these commands create a table, then create appropriate CMP indexes:

```
CREATE TABLE f(c1 INT NOT NULL, c2 INT NULL, c3 CHAR(5), c4 CHAR(5))
```

```
CREATE CMP INDEX c1c2cmp ON f(c1, c2)
```

This index is illegal because the columns indexed are not of the same data type, precision, and scale:

```
CREATE CMP INDEX c1c3cmp ON f(c1, c3)
```

8.2.2.2 Restrictions on CMP Indexes

The following restrictions apply to CMP.

- You can drop CMP indexes.
- CMP indexes cannot be unique.
- An exception is raised if you attempt to alter or delete a column that is defined in a CMP index.
- You cannot ALTER TABLE MODIFY an existing column that is defined in a CMP index.
- CMP indexes do not support the following data types:
  - BIT
  - FLOAT
  - DOUBLE
  - REAL

8.2.3 Date (DATE) and Datetime (DTTM) Indexes

DATE and DTTM index types process queries involving date or datetime quantities.

Index	Description
DATE	Used on columns of data type DATE to process certain queries involving date quantities.
DTTM	Used on columns of data type DATETIME or TIMESTAMP to process certain queries involving datetime quantities.

In this section:

Recommended Use of DATE and DTTM Index Types [page 162]
Use a DATE or DTTM index when the DATE, DATETIME, or TIMESTAMP column is used in queries containing date and time functions and operations.

Advantages and Disadvantages of DATE/DTTM [page 165]
Advantages and disadvantages of using a DATE or DTTM index.

Comparison to Other Indexes [page 166]
The DATE, TIME, and DTTM indexes have performance consistent with the HNG index.

Additional Indexes for DATE/DTTM Columns [page 166]
Always use a DATE, TIME, or DTTM index on a column of DATE, TIME, DATETIME, or TIMESTAMP data type, if the column is referenced in the WHERE clause, in ON conditions, or in the GROUP BY clause.

8.2.3.1 Recommended Use of DATE and DTTM Index Types

Use a DATE or DTTM index when the DATE, DATETIME, or TIMESTAMP column is used in queries containing date and time functions and operations.

- Queries with DATEPART equality predicates (=, !=), DATEPART range predicates (>, <, >=, <=, !>, !<, BETWEEN) and DATEPART IN list predicates
- Queries with range predicates (>, <, >=, <=, BETWEEN)

> **i Note**
>
> For a simple equality predicate (no DATEPART) with a DATE, DATETIME, or TIMESTAMP column, the HG index provides the best performance. If an HG index is unavailable, the DATE or DTTM index is used to get the result.
>
> If a DATE, DATETIME, or TIMESTAMP column is used in the GROUP BY clause or in the WHERE/HAVING clauses for equalities (including join conditions) or IN predicates, the column needs an HG index, as only these indexes can perform fast equality.

The table tab used in the examples contains columns defined as follows:

```
CREATE TABLE tab
 (col1 DATE,
  col2 DATETIME,
  col3 TIME);
```

In this section:

Queries with DATEPART Equality, Range, and IN List Predicates [page 162]
For a query with an equality predicate (= or !=), if one side of the comparison is a DATEPART expression or some other date and time function (for example, YEAR, QUARTER, DAY, MINUTE), and the other side of the comparison is a constant expression (including a constant value or host variable), then the DATE, TIME, or DTTM index is used (if the index is available) to get the result set.

Queries with Range Predicates [page 164]
Examine how your queries use range predicates when indexing predicate columns.

8.2.3.1.1 Queries with DATEPART Equality, Range, and IN List Predicates

For a query with an equality predicate (= or !=), if one side of the comparison is a DATEPART expression or some other date and time function (for example, YEAR, QUARTER, DAY, MINUTE), and the other side of the

comparison is a constant expression (including a constant value or host variable), then the DATE, TIME, or DTTM index is used (if the index is available) to get the result set.

For example, the DATE, TIME, or DTTM index is used in these queries:

```
SELECT * FROM tab WHERE DATEPART(YEAR, col1) = 2002;
SELECT * FROM tab WHERE DATEPART(HOUR, col2) = 20;
SELECT * FROM tab WHERE MINUTE (col3) != 30;
SELECT * FROM tab WHERE DATEPART(MONTH, col2) = @tmon;
```

In this example, @tmon is an INTEGER host variable.

The appropriate DATEPART range and IN list predicate conditions for processing with DATE, TIME, and DTTM indexes are:

- COMPARISON conditions >, <, >=, <=, !>, !<
  One side of the operator is a date/time function or DATEPART function, with a parameter that is a table column or view column. The other side of the operator is a constant expression, such as an integer or integer type host variable. For example:

  ```
  DATEPART(WEEK, col1) !<23
  ```

  ```
  DATEPART(YEAR, col1) = 2001
  ```

  ```
  HOUR(col3) >= 1
  ```

- BETWEEN ... AND condition
  The left side of BETWEEN is a date/time function or DATEPART function, with a parameter that is a table column or view column. Both sides of the AND are constant expressions, such as integers or integer type host variables. For example:

  ```
  DATEPART(YEAR, col1) BETWEEN host-var1 AND host-var2
  ```

- IN conditions
  The left side of IN is a date/time function or DATEPART function, with a parameter that is a table column or view column. The values inside the IN list are constant expressions. For example:

  ```
  DATEPART(MONTH, col1) IN (1999, 2001, 2003)
  ```

> **i Note**
>
> The DATE, TIME, and DTTM indexes do not support some date parts (Calyearofweek, Calweekofyear, Caldayofweek, Dayofyear, Millisecond). For example:
>
> ```
> SELECT * FROM tab WHERE DATEPART(MILLISECOND, col3)
> = 100;
> SELECT * FROM tab WHERE DATEPART(DAYOFYEAR, col1) <= 89;
> ```
>
> In these cases, the query optimizer chooses other indexes to get the result.

8.2.3.1.2 Queries with Range Predicates

Examine how your queries use range predicates when indexing predicate columns.

In the following cases with range predicates, a `DATE`, `TIME`, or `DTTM` index is chosen to process the queries:

- Compare condition:

  ```
  SELECT * FROM tab WHERE col1 < '2002/10/09';
  SELECT * FROM tab WHERE col2 >= '2002/01/01 09:12:04.006';
  ```

  One side of the comparison operator is a column name and the other side is a constant expression (constant value or host variable).
- Between condition:

  ```
  SELECT * FROM tab WHERE col3 BETWEEN '09:12:04.006' AND '20:12:04.006';
  SELECT * FROM tab WHERE col2 BETWEEN tmp_datetime1 AND tmp_datetime2;
  ```

For these types of queries, a `DATE`, `TIME`, or `DTTM` index is usually faster than a `HNG` index.

In three specific cases, use of the `DATE` or `DTTM` index may significantly improve performance:

- The range of the predicate is exactly one or more years (the actual start date is the beginning of a year and the actual end date is the end of a year). For example:

  ```
  SELECT * FROM tab WHERE col1 BETWEEN '1993-01-01' AND '1996-12-31';
  SELECT * FROM tab WHERE col1 >= '1993-01-01' AND
  col1 < '1997-01-01';
  SELECT * FROM tab WHERE col2 BETWEEN '1993-01-01 00:00:00.000000' AND
  '1996-12-31 23:59:59.999999';
  ```

- The range of the predicate is exactly one or more months in the same year (the actual start date is the beginning of a month and the actual end date is the end of a month). For example,

  ```
  SELECT * FROM tab WHERE col1 > '1993-01-31' AND
  col1 <= '1993-06-31';
  SELECT * FROM tab WHERE col2 >= '1993-01-01 00:00:00.000000' AND col1 <
  '1993-06-01 00:00:00.000000';
  ```

- The range of the predicate is exactly one day. For example,

  ```
  SELECT * FROM tab WHERE col2 >= '1993-01-31 00:00:00.000000' AND
  col2 <= '1993-01-31 23:59:59.999999';
  ```

> **i Note**
>
> In the three cases above, you must be careful about the concepts of range of years, range of months, and exactly one day. For example, there are four cases for a `DTTM` index that are recognized as range of years:
>
> ```
> col2 >  'year1/12/31 23:59:59.999999' and
> col2 <  'year2/01/01 00:00:00.000000'
> col2 >= 'year1/01/01 00:00:00.000000' and
> col2 <  'year2/01/01 00:00:00.000000'
> col2 >  'year1/12/31 23:59:59.999999' and
> col2 <= 'year2/12/31 23:59:59.999999'
> col2 >= 'year1/01/01 00:00:00.000000' and
> col2 <= 'year2/12/31 23:59:59.999999'
> ```

Ranges as in the following examples do not match range of years:

```
col2 > 'year1/12/31 23:59:59.999999' and
col2 <= 'year2/01/01 00:00:00.000000'
col2 > 'year1/01/01 00:00:00.000000' and
col2 <  'year2/01/01 00:00:00.000000'
```

The first range does not match, because it includes the value 'year2/01/01 00:00:00:000000' in addition to the range of years. The second range loses the value 'year1/01/01 00:00:00.000000.'

Similar specifics apply to range of months, and exactly one day, for both `DTTM` and `DATE` indexes.

If a small date range (less than 60 values) does not fit the three specific cases above, then `HG` indexes are faster than the `DATE` index.

8.2.3.2 Advantages and Disadvantages of DATE/DTTM

Advantages and disadvantages of using a `DATE` or `DTTM` index.

Advantages	Disadvantages
• Queries with date, time, or datetime quantities are resolved more quickly than with other index types. • You can create and drop a `DATE` or `DTTM` index.	• Uses more disk space than `HNG` index. • Fast equality still requires HG index. • You can use these indexes only if data in the column is `DATE`, `DATETIME`, or `TIMESTAMP` data type.

In this section:

Restrictions on DATE/DTTM Indexes [page 165]
Restrictions apply to `DATE` and `DTTM` indexes:

8.2.3.2.1 Restrictions on DATE/DTTM Indexes

Restrictions apply to `DATE` and `DTTM` indexes:

- Cannot use the UNIQUE keyword.
- Can only be created on a single column.
- Do not support date parts Calyearofweek, Calweekofyear, Caldayofweek, Dayofyear, Millisecond.

8.2.3.3 Comparison to Other Indexes

The DATE, TIME, and DTTM indexes have performance consistent with the HNG index.

Compared to HNG, DATE, TIME, and DTTM indexes are generally up to twice as fast as HNG in the supported cases. Therefore, an HNG index is not necessary in addition to a DATE, TIME, or DTTM index on a column of DATE, TIME, DATETIME, or TIMESTAMP data type.

8.2.3.4 Additional Indexes for DATE/DTTM Columns

Always use a DATE, TIME, or DTTM index on a column of DATE, TIME, DATETIME, or TIMESTAMP data type, if the column is referenced in the WHERE clause, in ON conditions, or in the GROUP BY clause.

In addition, the HG index may also be appropriate for a DATE, TIME, DATETIME, or TIMESTAMP column, especially if you are evaluating equality predicates against the column.

8.2.4 Fast Projection (FP) Index

By default, the database engine automatically creates a Fast Projection (FP) index on all columns. FP indexes optimize projections and allow the database engine to evaluate certain kinds of search conditions.

An FP index is an array of <n> fixed-length entries where <n> is the number of rows in the table. The IQ UNIQUE storage directive applied to the column determine whether the column loads as a flat FP or NBit. Flat FP indexes contain actual column cell values. NBit is a compression scheme that uses <n> bits to index the dictionary where the data is stored. All datatypes except LOB (both character and binary) and BIT datatypes may be NBit columns.

IQ UNIQUE

IQ UNIQUE defines the expected cardinality of a column and determines whether the column loads as Flat FP or NBit. An IQ UNIQUE(<n>) value explicitly set to 0 loads the column as Flat FP. Columns without an IQ UNIQUE constraint implicitly load as NBit up to the limits defined by the FP_NBIT_AUTOSIZE_LIMIT and FP_NBIT_LOOKUP_MB options:

- FP_NBIT_AUTOSIZE_LIMIT limits the number of distinct values that load as NBit.
- FP_NBIT_LOOKUP_MB sets a threshold for the total NBit dictionary size.
- FP_NBIT_ROLLOVER_MAX_MB sets the dictionary size for implicit NBit rollovers from NBit to Flat FP.
- FP_NBIT_ENFORCE_LIMITS enforces NBit dictionary sizing limits. This option is OFF by default.

Using IQ UNIQUE with an <n> value less than the FP_NBIT_AUTOSIZE_LIMIT is not necessary. Auto-size functionality automatically sizes all low or medium cardinality columns as NBit. Use IQ UNIQUE in cases where you want to load the column as Flat FP or when you want to load a column as NBit when the number of distinct values exceeds the FP_NBIT_AUTOSIZE_LIMIT.

Rebuilding FP Column Indexes

`sp_iqindexmetadata` displays details about column indexes, including the `FP` index type (`Flat FP` or `NBIT`), the distinct counts, `IQ UNIQUE <n>` value, and dictionary size.

`sp_iqrebuildindex` rebuilds `Flat FP` as `NBIT`, or `NBIT` as `Flat FP`. The `index_clause` can reset `IQ UNIQUE <n>` to an explicit value from '0' (to recast an `NBIT` column to `Flat FP`) up to the limits defined in the `FP_NBIT_AUTOSIZE_LIMIT` and `FP_NBIT_LOOKUP_MB` options. If the count exceeds the `<n>` value, and `FP_NBIT_ENFORCE_LIMITS=ON`, the operation rolls back. If the `FP_NBIT_ENFORCE_LIMITS=OFF` (default), the `NBIT` dictionary continues to grow.

Additional Information

- *SAP IQ SQL Reference > Database Options > Alphabetical List of Options*:
  - *FP_NBIT_AUTOSIZE_LIMIT*
  - *FP_NBIT_LOOKUP_MB*
  - *FP_NBIT_ROLLOVER_MAX_MB*
  - *FP_NBIT_ENFORCE_LIMIT*
- *SAP IQ SQL Reference > System Procedures > Alphabetical List of System Stored Procedures*:
  - *sp_iqrebuildindex*
  - *sp_iqindexmetadata*
- *SAP IQ Performance and Tuning Guide*:
  - *Structuring Queries > Structuring Queries > Zone Maps*
  - *Controlling Query Processing> Setting Query Optimization Options*

8.2.5 High_Group (HG) Index

The High_Group index is commonly used for join columns with integer data types. It is also more commonly used than High_Non_Group because it handles `GROUP BY` efficiently.

Primary key, unique, and non-unique single-column HG tiered and non-tiered indexes support CHAR, VARCHAR, BINARY, and VARBINARY datatypes to a maximum of 5300 bytes.

Multi-column HG indexes (both unique and non-unique) can contain columns with widths greater than 255 bytes up to a combined maximum width of 5300 bytes. The 5300 byte maximum includes overhead of 2 bytes per varbinary column and 1 byte per other data type column.

In this section:

Recommended Use of HG [page 168]
HG indexes and multicolumn HG indexes offer different functionality that must be considered when choosing which to use.

Advantages and Disadvantages of High_Group [page 168]
High_Group indexes have advantages and disadvantages.

Additional Indexes for HG Columns [page 169]
In some cases, a column that meets the criteria for a High_Group index may be used in queries where a different type of index may be faster. If this is the case, create additional indexes for that column.

Automatic Creation of High_Group Index [page 169]
SAP IQ creates a High_Group index by default whenever you issue a `CREATE INDEX` statement without specifying an index type.

8.2.5.1 Recommended Use of HG

HG indexes and multicolumn HG indexes offer different functionality that must be considered when choosing which to use.

Use an `HG` index when:

- The column will be used in a join predicate
- A column has more than 1000 unique values

Use multicolumn `HG` indexes to enhance the performance of `ORDER BY` queries with references to multiple columns. This change is transparent to users, but improves query performance.

> **i Note**
>
> Foreign key columns require their own, individual `HG` index.

8.2.5.2 Advantages and Disadvantages of High_Group

High_Group indexes have advantages and disadvantages.

Advantages	Disadvantages
• Quickly processes queries with `GROUP BY`.	• This index needs more disk space than the `HNG` index (it can take as much as three times more space than raw data). • This index type takes the longest time to populate with data, and to delete. • You cannot use this index if data in your columns is `BIT`. • This index is not recommended for `FLOAT`, `REAL`, and `DOUBLE` data.

8.2.5.3 Additional Indexes for HG Columns

In some cases, a column that meets the criteria for a High_Group index may be used in queries where a different type of index may be faster. If this is the case, create additional indexes for that column.

8.2.5.4 Automatic Creation of High_Group Index

SAP IQ creates a High_Group index by default whenever you issue a `CREATE INDEX` statement without specifying an index type.

SAP IQ automatically creates a High_Group index for any `UNIQUE`, `FOREIGN KEY`, or `PRIMARY KEY` constraint. For foreign keys of a single column, SAP IQ creates a single column non-unique High_Group index. For multicolumn foreign keys, a non-unique composite High_Group index is implicitly created. The non-unique HG index allows duplicate values and optionally allows nulls. It provides the building block for referential integrity and can be used to improve query performance.

SAP IQ allows the use of NULL in data values on a user created unique multicolumn `HG` index, if the column definition allows for NULL values and a constraint (primary key or unique) is not being enforced. For more details on Multicolumn Indexes, see *CREATE INDEX Statement* in *SAP IQ SQL Reference*.

Queries with joins on multiple columns or multicolumn group by clauses may improve performance because a non-unique composite High Group index provides more accurate cardinality estimates of joins and result sizes. It can also optimize pushdowns and subqueries.

8.2.6 High_Non_Group (HNG) Index

The `HNG` index was common for range searches. The `HG` index offers better performance.

We recommend migrating your `HNG` indexes to `HG` indexes.

8.2.7 TEXT Index

To use `TEXT` indexes, you must be specifically licensed to use the unstructured data analytics functionality.

Unlike a Containment (WD) index, which uses keywords in a column string, a `TEXT` index stores positional information for terms in the indexed columns. Queries that use `TEXT` indexes can be faster than those that must scan all the values in the table.

See *SAP IQ Administration: Unstructured Data Analytics*.

In this section:

Creating a TEXT Index Using Interactive SQL [page 170]
Before you can perform a full text search, you must create a `TEXT` index on the columns you want to search.

8.2.7.1 Creating a TEXT Index Using Interactive SQL

Before you can perform a full text search, you must create a `TEXT` index on the columns you want to search.

Context

A `TEXT` index stores positional information for terms in the indexed columns.

Procedure

1. Connect to the database as a user with CREATE ANY INDEX or CREATE ANY OBJECT system privilege, or as the owner of the underlying table.
2. Execute a `CREATE TEXT INDEX` statement.

Example

This example creates a `TEXT` index, `myTxtIdx`, on the `CompanyName` column of the `Customers` table in the `iqdemo` database. The `default_char` text configuration object is used:

```
CREATE TEXT INDEX myTxtIdx ON Customers
    ( CompanyName ) CONFIGURATION default_char
```

8.2.8 Time (TIME) Index

Use the TIME index to process queries involving time quantities.

The `TIME` index is used on columns of data type `TIME`.

In this section:

Recommended use of TIME Index Type [page 171]
Use a `TIME` when the `TIME` column is used in queries containing time functions and operations.

Advantages and Disadvantages of TIME [page 171]
`TIME` indexes have advantages and disadvantages.

8.2.8.1 Recommended use of TIME Index Type

Use a `TIME` when the `TIME` column is used in queries containing time functions and operations.

- Queries with DATEPART equality predicates (=, !=), DATEPART range predicates (>, <, >=, <=, !>, !<, `BETWEEN`) and DATEPART IN list predicates
- Queries with range predicates (>, <, >=, <=, `BETWEEN`)

> **i** Note
>
> For a simple equality predicate (no DATEPART) with a `TIME` column, `HG` indexes have the best performance. If an `HG` index is unavailable, the`TIME` index gets the result.
>
> If a `TIME` or `TIMESTAMP` column is used in the GROUP BY clause or in the WHERE/HAVING clauses for equalities (including join conditions) or IN predicates, the column needs an `HG` index, as only these indexes can do fast equality.

8.2.8.2 Advantages and Disadvantages of TIME

`TIME` indexes have advantages and disadvantages.

Advantages	Disadvantages
• Queries with time quantities are resolved more quickly than with other index types. • You can create and drop a `TIME` index.	• Uses more disk space than `HNG` index. • Fast equality still requires `HG` index. • You can use this index only if data in the column is `TIME` data type.

In this section:

Restrictions on TIME Indexes [page 171]
`TIME` indexes have some functional restrictions.

8.2.8.2.1 Restrictions on TIME Indexes

`TIME` indexes have some functional restrictions.

The restrictions are:

- Cannot use the UNIQUE keyword.
- Can only be created on a single column.
- Does not support date part Millisecond.

8.3 Criteria for Choosing Indexes

The set of indexes that you define for any given column can have dramatic impact on the speed of query processing.

There are four main criteria for choosing indexes:

- Number of unique values
- Types of queries
- Disk space usage
- Data types

Use the recommendations for all criteria in combination, rather than individually. Remember also that all columns are automatically stored in a way that facilitates fast projections. To decide on additional indexes, look closely at the data in each column. Try to anticipate the number of unique and total values expected in query results.

In this section:

Number of Unique Values in the Index [page 172]
: Indexes are optimized according to the number of unique (distinct) values they include.

Types of Queries [page 173]
: Certain index types optimize particular types of queries.

Indexes Recommended by Query Type [page 174]
: Use the index type recommended for your query.

Indexes Recommended by Column Value [page 174]
: Criteria such as the number of unique values in a column help identify appropriate index types for data.

Disk Space Usage [page 175]
: Estimates of the amount of space each index uses compared to the amount of column data from the source database or flat file.

Data Types in the Index [page 176]
: When indexing a column, choose only supported index types for the column data type.

Index Type Combinations [page 177]
: When a column is used in more than one type of query, you might want to assign more than one column index type.

8.3.1 Number of Unique Values in the Index

Indexes are optimized according to the number of unique (distinct) values they include.

`IQ UNIQUE` defines the expected cardinality of a column and determines whether the column loads as `Flat FP` or `NBit`. An `IQ UNIQUE(<n>)` value explicitly set to 0 loads the column as `Flat FP`. Columns without an IQ UNIQUE constraint implicitly load as `NBit` up to the limits defined by the `FP_NBIT_AUTOSIZE_LIMIT` and `FP_NBIT_LOOKUP_MB` options:

- FP_NBIT_AUTOSIZE_LIMIT limits the number of distinct values that load as NBit.
- FP_NBIT_LOOKUP_MB sets a threshold for the total NBit dictionary size.
- FP_NBIT_ROLLOVER_MAX_MB sets the dictionary size for implicit NBit rollovers from NBit to Flat FP.
- FP_NBIT_ENFORCE_LIMITS enforces NBit dictionary sizing limits. This option is OFF by default.

Using IQ UNIQUE with an <n> value less than the FP_NBIT_AUTOSIZE_LIMIT is not necessary. Auto-size functionality automatically sizes all low or medium cardinality columns as NBit. Use IQ UNIQUE in cases where you want to load the column as Flat FP or when you want to load a column as NBit when the number of distinct values exceeds the FP_NBIT_AUTOSIZE_LIMIT.

Regardless of the number of distinct values, choose the HG index. HG offers better performance than HNG.

8.3.2 Types of Queries

Certain index types optimize particular types of queries.

Determine in advance how data in the columns will generally be queried. For example:

- Will the column be part of a join predicate?
- If the column has a high number of unique values, will the column be used in a GROUP BY clause, be the argument of a COUNT DISTINCT, or be in the SELECT DISTINCT projection?
- Will the column frequently be compared with another column of the same data type, precision, and scale?

Often, the type of data in a column gives a good indication how the column will be used. For example, a date column will probably be used for range searches in WHERE clauses, and a column that contains prices or sales amounts will probably be used in the projection as an argument for aggregate functions (SUM, AVG, and so on).

> **i Note**
>
> SAP IQ can still resolve queries involving a column indexed with the wrong index type, although it may not do so as efficiently.

For optimal query performance, columns used in join predicates, subquery predicates, GROUP BY and DISTINCT clauses should have an HG index, since SAP IQ has no statistics other than the index for the optimizer to use.

These estimates are generally valid; however, other factors can take precedence:

- For range predicates, the number of unique values is the most important factor.
- When using the set functions COUNT, COUNT DISTINCT, SUM, MIN, MAX, and AVG, to use any index other than the default, the entire query must be resolvable using a single table.
- BIT data can be used only in the default index; VARBINARY data greater than 255 bytes can be used only in the default, TEXT, and CMP index types; CHAR and VARCHAR data greater than 255 bytes can be used only in the default, CMP, TEXT, and WD index types; LONG VARCHAR data can be used only in the default, TEXT, and WD index types; only DATE data can be used in the DATE index type; only TIME data can be used in the TIME index type; only DATETIME and TIMESTAMP data can be used in the DTTM index type.

8.3.3 Indexes Recommended by Query Type

Use the index type recommended for your query.

The index that is usually fastest for each query is listed first, the slowest last. These recommendations should not be your only criteria for choosing an index type; consider the number of unique values and disk space.

Type of Query Usage	Recommended Index Type
In a `SELECT` projection list	Default
In calculation expressions such as `SUM`(A+B)	Default
As `AVG`/`SUM` argument	`HG`, default
As `MIN`/`MAX` argument	`HG`
As `COUNT` argument	Default
As `COUNT DISTINCT, SELECT DISTINCT` or `GROUP BY` argument	`HG`, default
As analytical function argument	default
If field does not allow duplicates	`HG`
Columns used in ad hoc join condition	Default, `HG`
As `LIKE` argument in a `WHERE` clause	Default
As `IN` argument	`HG`
In equality or inequality (=, !=)	`HG`; also `CMP`
In range predicate in `WHERE` clause (>, <, >=, <=, `BETWEEN`	`HG`; also `CMP`, `DATE`, `TIME`, `DTTM`
In `DATEPART` equality, range, and IN list predicates	`DATE`, `TIME`, `DTTM`
In a `CONTAINS` predicate	`WD`, `TEXT`

8.3.4 Indexes Recommended by Column Value

Criteria such as the number of unique values in a column help identify appropriate index types for data.

Criteria to Identify	Index to Select
Note indexes created automatically on all columns.	Default index
Note indexes created automatically on columns with `UNIQUE` or `PRIMARY KEY` constraint.	`HG` with `UNIQUE` enforced
Identify all columns used in a join predicate and choose the index type depending on the number of unique values.	`HG`
Identify columns that contain a low number of unique values and do not already use multiple indexes.	`HG`

Criteria to Identify	Index to Select
Identify columns that have a high number of unique values and that are part of a `GROUP BY` clause in a select list in a `SELECT DISTINCT` or `DISTINCT COUNT`.	`HG`
Identify columns that may be used in the `WHERE` clause of ad hoc join queries that do not already have an `HG` index.	`HG`
Identify columns that have a high number of unique values and that will not be used with `GROUP BY`, `SELECT DISTINCT` or `DISTINCT COUNT`.	`HG`
Identify pairs of columns with the same data type, precision, and scale that are likely to need frequent comparison.	`CMP`
Identify columns that contain a list of keywords or a URL.	`WD`
Identify columns of `DATE`, `TIME`, `DATETIME`, or `TIMESTAMP` that have a high number of unique values and that will not be used with `GROUP BY`, `SELECT DISTINCT`, or `DISTINCT COUNT`.	`DATE`, `TIME`, or `DTTM`
Look at any remaining columns and decide on additional indexes based on the number of unique values, type of query, and disk space. Also, for all columns, be sure that the index types you select allow the data type for that column.	

8.3.5 Disk Space Usage

Estimates of the amount of space each index uses compared to the amount of column data from the source database or flat file.

Type of Index	Estimated Space Versus Raw Data
Default	Smaller than or equal to If the number of distinct values is less than 255, this index uses significantly less space than the raw data.
High_Group	Smaller than up to twice as large As the number of distinct values decreases (that is, the number of entries per group increases), the space used decreases in proportion to the size of the raw data. For `HG` indexes, the index size depends on the number of unique values. The more unique values, the more space the index requires.
Date	Smaller than or equal to
Time	Smaller than or equal to
Datetime	Smaller than or equal to

> **i** Note
>
> Compare (CMP) indexes do not save disk space because they are always an additional index.

8.3.6 Data Types in the Index

When indexing a column, choose only supported index types for the column data type.

The default index allows any data type.

Data Type	Supported Indexes	Unsupported Indexes
tinyint	CMP, HG (HNG, LF permitted but not recommended)	WD, DATE, TIME, DTTM, TEXT
smallint	CMP, HG (HNG, LF permitted but not recommended)	WD, DATE, TIME, DTTM, TEXT
int	CMP, HG (HNG, LF permitted but not recommended)	WD, DATE, TIME, DTTM, TEXT
unsigned int	CMP, HG (HNG, LF permitted but not recommended)	WD, DATE, TIME, DTTM, TEXT
bigint	CMP, HG (HNG, LF permitted but not recommended)	WD, DATE, TIME, DTTM, TEXT
unsigned bigint	CMP, HG (HNG, LF permitted but not recommended)	WD, DATE, TIME, DTTM, TEXT
numeric, decimal	CMP, HG (HNG, LF permitted but not recommended)	WD, DATE, TIME, DTTM, TEXT
double	HG (LF permitted but not recommended)	CMP, HNG, WD, DATE, TIME, DTTM, TEXT
float	HG (LF permitted but not recommended)	CMP, HNG, WD, DATE, TIME, DTTM, TEXT
real	HG (LF permitted but not recommended)	CMP, HNG, WD, DATE, TIME, DTTM, TEXT
bit	(Default index only)	CMP, HG, HNG, LF, WD, DATE, TIME, DTTM, TEXT
date	CMP, HG, DATE	WD, TIME, DTTM, TEXT
time	CMP, HG, TIME	WD, DATE, DTTM, TEXT
datetime, timestamp	CMP, HG, DTTM	WD, DATE, TIME, TEXT
char <= 255 bytes, character	CMP, HG, WD, TEXT	DATE, TIME, DTTM
char >255 bytes	CMP, WD, TEXT	HG, HNG, LF, DATE, TIME, DTTM
varchar <= 255 bytes	CMP, HG, WD, TEXT	DATE, TIME, DTTM
varchar >255 bytes	CMP, WD, TEXT	HG, HNG, LF, DATE, TIME, DTTM
long varchar	WD, TEXT	CMP, HG, HNG, LF, DATE, TIME, DTTM
binary <= 255 bytes	CMP, HG, TEXT	HNG, WD, DATE, TIME, DTTM
CMP, TEXT	HG, HNG, LF, WD, DATE, TIME, DTTM	

Data Type	Supported Indexes	Unsupported Indexes
varbinary <= 255 bytes	CMP, HG, TEXT	HNG, WD, DATE, TIME, DTTM
varbinary > 255 bytes	CMP, TEXT	HG, HNG, LF, WD, DATE, TIME, DTTM

8.3.7 Index Type Combinations

When a column is used in more than one type of query, you might want to assign more than one column index type.

	Index to Add			
Existing Index	HG	CMP[a]	WD	DATE, TIME, or DTTM
HG	-	1	1	1

1 = Combination recommended.

a. A CMP index applies to a pair of columns. Each of those columns always has at least one other index.

8.4 Creating Indexes

SAP IQ automatically creates a default index on every table column when you create a database in an IQ store. For best query performance, create additional indexes.

Prerequisites

If you are creating an index on a partitioned table, all dbspaces must be online.

Procedure

1. Choose the right index type for your data. Make sure that the selected index type allows the data type for that column.

   If you do not specify an index type, SAP IQ creates an `HG` index. Several front-end tools create an `HG` index automatically for this reason.
2. Create an index using any method:
   - Enter a `CREATE INDEX` statement.

For example, create a High_Non_Group (HNG) index called ShipIx on the ShipDate column of the SalesOrderItems table.

```
CREATE HNG INDEX ShipIx
  ON dbo.SalesOrderItems (ShipDate)
```

- Use the SAP IQ Cockpit Create Index wizard.
- Use a PRIMARY KEY, or UNIQUE column constraint of CREATE TABLE, which creates an HG index automatically.

In this section:

Concurrent Column Indexing [page 178]
In some cases, you can create more than one column index simultaneously.

Status Messages for Index Loading [page 178]
If you enable progress messages, status messages display after every 100,000 records are inserted.

Adding Column Indexes After Inserting Data [page 179]
When you create an additional column index, the CREATE INDEX command creates the new index as part of the individual table.

Executing Groups of CREATE INDEX Statements [page 179]
Use the keywords BEGIN PARALLEL IQ and END PARALLEL IQ to delimit any number of CREATE INDEX statements that you want to execute as a group at the same time.

8.4.1 Concurrent Column Indexing

In some cases, you can create more than one column index simultaneously.

- Each CREATE INDEX statement can create only one index.
- If two connections issue CREATE INDEX statements on the same table, the first statement works; the second receives an error stating that only one writer is allowed.
- If two connections issue CREATE INDEX statements on different tables, both proceed in parallel.

8.4.2 Status Messages for Index Loading

If you enable progress messages, status messages display after every 100,000 records are inserted.

To enable messages, set the NOTIFY option of CREATE INDEX to a non-zero value.

To change the number of records, use the NOTIFY option of CREATE INDEX, or the option NOTIFY_MODULUS. To prevent these messages, specify NOTIFY 0.

8.4.3 Adding Column Indexes After Inserting Data

When you create an additional column index, the `CREATE INDEX` command creates the new index as part of the individual table.

If the existing column indexes in the individual table already contain data, the `CREATE INDEX` statement also inserts data into the new index from an existing index. This ensures data integrity among all the column indexes for columns within an individual table.

This capability is useful if you discover that a column needs an additional index after you have already inserted data. This allows you to add the index without having to start over.

> **i** Note
>
> Inserting data from an existing index can be slow. It is always faster to create all the appropriate indexes before you insert data, then insert into all of them at once, with either the `LOAD TABLE` or `INSERT` statement.

8.4.4 Executing Groups of CREATE INDEX Statements

Use the keywords `BEGIN PARALLEL IQ` and `END PARALLEL IQ` to delimit any number of `CREATE INDEX` statements that you want to execute as a group at the same time.

Use these keywords only to create indexes on base tables in the IQ main store, not temporary tables or catalog store tables.

If any of these `CREATE INDEX` statements fails, they all roll back:

```
BEGIN PARALLEL IQ
    CREATE HG INDEX c1_HG on table1 (col1);
    CREATE HNG INDEX c12_HNG on table1 (col12);
    CREATE HNG INDEX c2_HNG on table1 (col2);
END PARALLEL IQ
```

8.5 Running the Index Advisor

If you set the INDEX_ADVISOR option on your database, SAP IQ issues messages in the message log or query plan to suggest additional indexes that might improve performance.

Procedure

1. In Interactive SQL, set the `INDEX_ADVISOR` option:

   ```
   SET OPTION index_advisor = 'ON'
   ```

2. Examine the message log file. By default, the message log is named `dbname.iqmsg`, and the file is created in the same directory as the catalog store.

   Look for messages that focus on the following areas:
   - Local predicate columns
   - Single-column join key columns
   - Correlated subquery columns
   - Grouping columns
3. If you decide to follow the recommendations of the index advisor messages, create the suggested indexes:

```
CREATE HG INDEX id_hg
ON SalesOrderItems
( ID ) IN Dsp5
```

8.6 Renaming Indexes

You can rename an index in a base table or global temporary table with the owner type USER.

Use the ALTER INDEX statement to rename an index.

> **i** Note
>
> You cannot rename indexes created to enforce key constraints.

8.7 Viewing Indexes

To see information about indexes, use the `sp_iqindex` stored procedure.

To display information for all tables, omit the table name from `sp_iqindex`:

```
sp_iqindex
```

The following is example output:

index_name	unique_index	dbspace_id	remarks
ASIQ_IDX_T740_C3_FP	N	16,387	(NULL)
ASIQ_IDX_T740_C3_HG	N	16,387	(NULL)
ASIQ_IDX_T740_C1_FP	U	16,387	(NULL)
ASIQ_IDX_T740_I4_HG	Y	16,387	(NULL)
ASIQ_IDX_T740_C2_FP	N	16,387	(NULL)

The following example lists the indexes in the `Departments` table:

```
sp_iqindex 'Departments'
```

table_name	table_owner	column_name	index_type
Departments	GROUPO	DepartmentHeadID	FP
Departments	GROUPO	DepartmentHeadID	HG
Departments	GROUPO	DepartmentID	FP
Departments	GROUPO	DepartmentID	HG
Departments	GROUPO	DepartmentName	FP

In this section:

Index Information in System Views [page 181]
Information on indexes is in the consolidated view `SYSINDEXES` and the system view `SYSIQIDX`.

8.7.1 Index Information in System Views

Information on indexes is in the consolidated view `SYSINDEXES` and the system view `SYSIQIDX`.

See *SAP IQ SQL Reference* > *System Tables and Views* for a description of these views.

8.8 Removing Indexes

Remove column indexes that are no longer required.

Procedure

- Use either of the following methods:
  - Enter a `DROP` statement. See *DROP statement* in *SAP IQ SQL Reference*.
  - Delete the Index in SAP IQ.

8.9 Retaining Indexes When Removing Foreign Key Constraints

You may want to remove a foreign key constraint, but retain the underlying `HG` index. A non-unique `HG` index can provide query performance improvement, but may be expensive to build.

Procedure

Enter an `ALTER TABLE DROP FOREIGN KEY` statement.

See the `ALTER INDEX` statement in *SAP IQ SQL Reference*.

> **i** Note
>
> `ALTER TABLE DROP FOREIGN KEY CONSTRAINT` does not remove the automatically created nonunique `HG` index. You cannot drop a primary key if associated foreign keys remain. To remove such an index, drop it explicitly after issuing `ALTER TABLE DROP FOREIGN KEY`.

8.10 Optimizing Performance for Joins

You can create indexes to optimize join performance.

To gain the fastest processing of joins, create a High_Group (`HG`) index on all columns that may be referenced in:

- `WHERE` clauses of join queries
- `HAVING` clause conditions of join queries outside of aggregate functions

For example:

```
SELECT n_name, sum(l_extendedprice*(1-l_discount))
    AS revenue
    FROM customer, orders, lineitem, supplier,
        nation, region
    WHERE c_custkey        = o_custkey
        AND o_orderkey     = l_orderkey
        AND l_suppkey      = s_suppkey
        AND c_nationkey    = s_nationkey
        AND s_nationkey    = n_nationkey
        AND n_regionkey    = r_regionkey
        AND r_name         = 'ASIA'
        AND o_orderdate   >= '1994-01-01'
        AND o_orderdate    < '1995-01-01'
GROUP BY n_name
HAVING n_name LIKE "I%"
    AND SUM(l_extendedprice*(1-l_discount)) > 0.50
ORDER BY 2 DESC
```

All columns referenced in this query except `l_extendedprice` and `l_discount` should have an `HG` index.

9 Enforce Data Integrity

Table and column constraints and appropriate data type selection ensure that the data in your database is valid and reliable.

In this section:

Data Integrity Overview [page 184]
For data to have integrity means that the data is valid — that is, correct and accurate — and that the relational structure of the database is intact.

Column Defaults Encourage Data Integrity [page 187]
Column defaults automatically assign a specific value to a particular column or set of columns whenever a new row is entered into a database table.

Table and Column Constraints [page 195]
Constraints help to ensure that the data entered into a table is correct, and provide information to SAP IQ that boosts performance.

Entity and Referential Integrity [page 199]
Once you specify the primary key for each table, no further action is needed by client application developers or the database administrator to maintain entity integrity.

Integrity Rules in System Tables [page 207]
All the information about integrity checks and rules in a database is held in system tables and views.

Validating Catalog Store Indexes [page 208]
You can validate an index on SAP SQL Anywhere tables in the catalog store to ensure that every row referenced in the index actually works in the table.

9.1 Data Integrity Overview

For data to have integrity means that the data is valid — that is, correct and accurate — and that the relational structure of the database is intact.

The relational structure of the database is described through referential integrity constraints, which are business rules that maintain the consistency of data between tables.

SAP IQ supports stored procedures and JDBC, which allow you detailed control over how data gets entered into the database. For information on JDBC, see the *SAP IQ Programming Reference* guide.

In this section:

How Data Can Become Invalid [page 185]
Data may become invalid without proper checks.

Rules and Checks for Valid Data [page 185]
To help ensure that the data in a database is valid, formulate checks that define valid and invalid data, and design the rules to which data must adhere.

Statements That Change Database Contents [page 186]
Client applications change information in database tables by submitting SQL statements.

Data Integrity Tools [page 186]
Use data constraints and constraints that specify the referential structure of the database to maintain data integrity.

Statements that Implement Integrity Constraints [page 187]
Implement integrity constraints by submitting SQL statements.

9.1.1 How Data Can Become Invalid

Data may become invalid without proper checks.

Example scenarios that could cause data to become invalid:

- An operator enters orders to an orders table for a `customer_id` that does not exist in the customer table.
- An operator enters text where numeric data is required.
- An operator enters numeric data that is too wide for the column.
- More than one person enters the same information about a new department, with `dept_id` 200, to the department table.

9.1.2 Rules and Checks for Valid Data

To help ensure that the data in a database is valid, formulate checks that define valid and invalid data, and design the rules to which data must adhere.

Such rules are often called business rules. The collective name for checks and rules is constraints. Rules that maintain data integrity for a given column are column constraints. Rules that maintain integrity for one or more columns for a given table are table constraints. You can apply both table and column constraints to a single column in a table. Table constraints can also set the rule for a set of columns in a table.

Constraints that are built into the database itself are inherently more reliable than those built into client applications, or those spelled out as instructions to database users. Constraints built into the database are part of the database definition and can be enforced consistently across all applications.

Setting a constraint once, in the database, imposes it for all subsequent interactions with the database. By contrast, constraints that are built into client applications are vulnerable every time the software is altered, and may need to be imposed in several applications, or in several places in a single client application.

Declare any constraints that apply, whether or not SAP IQ enforces them. Declaring constraints ensures that you understand your data requirements, and are designing a database that matches the business rules of your organization.

SAP IQ performs several types of optimization based on the constraints you specify. This optimization does not depend on enforcement of constraints. For the best performance of queries and load operations, put all constraints in the database. `FOREIGN KEY`, `PRIMARY KEY` , `UNIQUE`, and `IQ UNIQUE` column constraints can improve load and query performance.

SAP IQ checks during load operations that certain constraints are obeyed. For example, SAP IQ ensures that data being loaded is the appropriate data type and length.

9.1.3 Statements That Change Database Contents

Client applications change information in database tables by submitting SQL statements.

Only a few SQL statements actually modify the information in a database:

- To delete an existing row of a table, use the `DELETE` statement.
- To insert a new row into a table, use the `INSERT` or `LOAD TABLE` statement.
- To change the value in a cell, use the `UPDATE` statement.

9.1.4 Data Integrity Tools

Use data constraints and constraints that specify the referential structure of the database to maintain data integrity.

Constraints

There are constraint types you can use on the data in individual columns or tables:

- A NOT NULL constraint prevents a column from containing a null entry. SAP IQ enforces this constraint.
- Columns can have CHECK conditions assigned to them, to specify that a particular condition should be met by every row for that column, for example, that salary column entries must be within a specified range.
- CHECK conditions can be made on the relative values in different columns, to specify, for example, in a library database that a `date_returned` entry is later than a `date_borrowed` entry.

Column constraints can be inherited from user-defined data types.

Entity and Referential Integrity

The information in relational database tables is tied together by the relations between tables. These relations are defined by the candidate keys and foreign keys that are built into the database design.

A foreign key is made up of a column or a combination of columns. Each foreign key relates the information in one table (the foreign table) to information in another (referenced or primary) table. A particular column, or combination of columns, in a foreign table is designated as a foreign key to the primary table.

The primary key or column (or set of columns) with a unique constraint is known as a candidate key. The referenced column or set of columns must be a candidate key and is called the referenced key. You cannot specify a foreign key constraint to a candidate key that is also a foreign key.

These referential integrity rules define the structure of the database:

- Keep track of the primary keys, guaranteeing that every row of a given table can be uniquely identified by a primary key that guarantees no nulls.
- Keep track of the foreign keys that define the relationships between tables. All foreign key values must either match a value in the corresponding primary key, if they are defined to allow NULL, or contain the NULL value.

9.1.5 Statements that Implement Integrity Constraints

Implement integrity constraints by submitting SQL statements.

- The `CREATE DATABASE` statement implements integrity constraints as the database is being created.
- The `ALTER DATABASE` statement adds integrity constraints, or deletes constraints, from an existing database.

9.2 Column Defaults Encourage Data Integrity

Column defaults automatically assign a specific value to a particular column or set of columns whenever a new row is entered into a database table.

A column default requires no action on the part of the client application. However, if the client application does specify a value for the column, the new value overrides the column default value.

Column defaults can quickly and automatically fill columns with information, such as the date or time a row is inserted or the user ID of the person who first modified a row in a table. Using column defaults encourages data integrity, but does not enforce it. Client applications can always override defaults.

In this section:

Supported Default Values [page 188]
SAP IQ supports default values for columns.

Default Value Restrictions [page 189]
Some column default values are not supported.

Creating Column Defaults [page 189]
You can use the `CREATE TABLE` statement to create column defaults when a table is created, or the `ALTER TABLE` statement to add column defaults later.

Changing Column Defaults [page 190]
To change column defaults, use the same form of the `ALTER TABLE` statement you use to create defaults.

Deleting Column Defaults [page 190]
To remove column defaults, modify them to NULL.

Supported Column Default Values [page 190]
You can load and insert column default values in SAP IQ.

Date, Time, and Timestamp Defaults [page 191]
: For columns with the `DATE`, `TIME`, or `TIMESTAMP` data type, you can use the `CURRENT DATE`, `CURRENT TIME`, `TIMESTAMP`, or `CURRENT TIMESTAMP` special value as a default. The default you choose must be compatible with the data type of the column.

USER Defaults [page 192]
: Assigning a `DEFAULT USER` to a column identifies the person making an entry in a database. This information may be required; for example, when salespeople are working on commission.

The IDENTITY or AUTOINCREMENT Default [page 193]
: The `IDENTITY/AUTOINCREMENT` default is useful for numeric data fields where the value of the number itself may have no meaning.

The NEWID Default [page 194]
: Use UUIDs (Universally Unique IDentifiers), also known as GUIDs (Globally Unique IDentifiers) to uniquely identify rows in a table.

The NULL Default [page 194]
: For columns that allow NULL values, specifying a NULL default is exactly the same as not specifying a default.

String and Number Defaults [page 194]
: You can specify a specific string or number as a default value, as long as the column holds a string or number data type.

Constant Expression Defaults [page 194]
: You can use a constant expression as a default value, as long as the expression does not reference database objects.

9.2.1 Supported Default Values

SAP IQ supports default values for columns.

- A string specified in the `CREATE TABLE` statement or `ALTER TABLE` statement
- A number specified in the `CREATE TABLE` statement or `ALTER TABLE` statement
- An automatically incremented number: one more than the previous highest value in the column
- UUID (Universally Unique Identifier) values generated by the `NEWID` function
- The current date, time, or timestamp
- The name of the current database
- The current user ID of the database user and the name of the user who last modified the row
- The publisher user ID of the database for SQL Remote applications
- A NULL value
- A constant expression, as long as it does not reference database objects
- A supported default value specified in a user-defined domain (data type) using the `CREATE DOMAIN` statement

9.2.2 Default Value Restrictions

Some column default values are not supported.

SAP IQ does not support the following values for column defaults:

- Values that use the special values `UTC TIMESTAMP`, `CURRENT UTC TIMESTAMP`, and `GLOBAL AUTOINCREMENT`
- A default value that is not compatible with the data type of the column
- A default value that violates the check constraint of the table or column
- A constant expression that references database objects

SAP IQ ignores settings for the `DEFAULT_TIMESTAMP_INCREMENT` database option.

9.2.3 Creating Column Defaults

You can use the `CREATE TABLE` statement to create column defaults when a table is created, or the `ALTER TABLE` statement to add column defaults later.

Context

You can also specify a default value when creating a user-defined domain (data type) using the `CREATE DOMAIN` statement.

The stored procedure `sp_iqcolumn` returns information about all columns for all tables. One of the column returned by the result set of `sp_iqcolumn` is called "default," and shows the particular default value for that column.

Procedure

1. Create a table named `tab1` with the default special value `LAST USER` specified for the `CHARACTER` column `c1`:

   ```
   CREATE TABLE tab1(c1 CHAR(20) DEFAULT LAST USER)
   ```

2. Add a condition to an existing column named `id` in the `sales_order` table, so that the value of the column automatically increments (unless a client application specifies a value):

   ```
   ALTER TABLE sales_order MODIFY id DEFAULT AUTOINCREMENT
   ```

3. Define a domain named `dom1` with a data type of `INTEGER` and a default value of 45:

   ```
   CREATE DOMAIN dom1 INTEGER DEFAULT 45
   ```

9.2.4 Changing Column Defaults

To change column defaults, use the same form of the `ALTER TABLE` statement you use to create defaults.

For example, to change the default value of a column named `order_date` from its current setting to `CURRENT DATE`, use:

```
ALTER TABLE sales_order
   MODIFY order_date DEFAULT CURRENT DATE
```

9.2.5 Deleting Column Defaults

To remove column defaults, modify them to NULL.

Procedure

Use the ALTER TABLE statement to remove column defaults.

This statement removes the default from the `order_date` column

```
ALTER TABLE sales_order
MODIFY order_date DEFAULT NULL
```

9.2.6 Supported Column Default Values

You can load and insert column default values in SAP IQ.

Use these statements:

- `INSERT...VALUES`
- `INSERT...SELECT`
- `INSERT...LOCATION`
- `LOAD TABLE`
- `UPDATE`
- `SELECT...FROM...FOR UPDATE`

When you are defining and inserting column default values:

- You can specify default values that cannot be evaluated by SAP IQ. An error is reported when an `INSERT`, `LOAD`, or `ALTER ADD` operation is performed on a table that has an unsupported default value.
- SAP IQ generates an error or warning when the server attempts to insert a default value that is incompatible with the data type of the column. For example, if you define a default expression of 'N/A' to an integer column, then any insert or load that does not specify the column value generates an error or warning, depending on the setting of the `CONVERSION_ERROR` database option.

- If a default value is too long for a CHARACTER type column, SAP IQ either truncates the string or generates an exception, depending on the setting of the STRING_RTRUNCATION database option.
- If the default value for a VARCHAR or LONG VARCHAR column is the zero-length string, SAP IQ inserts either a NULL or zero-length string, depending on the setting of the NON_ANSI_NULL_VARCHAR database option.
- If the default value for a VARCHAR, CHAR, or LONG VARCHAR column contains a partial multibyte character, SAP IQ may trim that character before inserting the value, depending on the setting of the TRIM_PARTIAL_MBC database option.
- If the default value violates the check constraint of either the table or the column, SAP IQ generates an error message every time the server attempts to insert the default value.
- Constraint violations that occur during a LOAD TABLE operation as a result of inserting default values are never ignored, regardless of any user-specified IGNORE CONSTRAINT and MESSAGE LOG/ROW LOG option.
- Column default values of UTC TIMESTAMP and CURRENT UTC TIMESTMAP are not supported by SAP IQ.
- Column default values defined on base tables are not propagated to the joins in which these tables participate.
- If a column on which a default value is defined is added to a table, all rows of the new column are populated with that default value.
- Changing the default value of an existing column in a table does not change any existing values in the table.
- The LOAD TABLE DEFAULTS option must be On to use the default value specified in the LOAD TABLE statement DEFAULT option. If the DEFAULTS option is Off, the specified load default value is not used and a NULL value is inserted into the column instead.
- The LOAD TABLE DEFAULT specification must contain at least one column to be loaded from the file specified in the LOAD TABLE command.
- The LOAD TABLE DEFAULT <default-value> must be of the same character set as that of the database, and must conform to the supported default values for columns and default value restrictions. The LOAD TABLE DEFAULT option does not support AUTOINCREMENT, IDENTITY, or GLOBAL AUTOINCREMENT as a load default value.
- Encryption of the default value is not supported for the load default values specified in the LOAD TABLE DEFAULT clause.

9.2.7 Date, Time, and Timestamp Defaults

For columns with the DATE, TIME, or TIMESTAMP data type, you can use the CURRENT DATE, CURRENT TIME, TIMESTAMP, or CURRENT TIMESTAMP special value as a default. The default you choose must be compatible with the data type of the column.

CURRENT DATE Default

A CURRENT DATE default might be useful to record:

- Dates of phone calls in a contact database
- Dates of orders in a sales entry database
- The date a patron borrows a book in a library database

CURRENT TIMESTAMP Default

The CURRENT TIMESTAMP is similar to the CURRENT DATE default, but offers greater accuracy. For example, a user of a contact management application may have several contacts with a single customer in one day; the CURRENT TIMESTAMP default is useful for distinguishing between these contacts.

Since CURRENT TIMESTAMP records a date and the time down to a precision of millionths of a second, you may also find CURRENT TIMESTAMP useful when the sequence of events is important in a database.

TIMESTAMP Default

When a column is declared with DEFAULT TIMESTAMP, a default value is provided for insert and load operations. The value is updated with the current date and time whenever the row is updated.

On INSERT and LOAD, DEFAULT TIMESTAMP has the same effect as CURRENT TIMESTAMP. On UPDATE, if a column with a default value of TIMESTAMP is not explicitly modified, the value of the column is changed to the current date and time.

SAP IQ does not support default values of UTC TIMESTAMP or CURRENT UTC TIMESTAMP, nor does SAP IQ support the database option DEFAULT_TIMESTAMP_INCREMENT. SAP IQ generates an error every time an attempt is made to insert or update the default value of a column of type UTC TIMESTAMP or CURRENT UTC TIMESTAMP.

9.2.8 USER Defaults

Assigning a DEFAULT USER to a column identifies the person making an entry in a database. This information may be required; for example, when salespeople are working on commission.

Building a user ID default into the primary key of a table is a useful technique for occasionally connected users, and helps prevent conflicts during information updates. These users can make a copy of the tables that are relevant to their work on a portable computer, make changes while not connected to a multiuser database, and then apply the transaction log to the server when they return.

USER Default

The special values USER and CURRENT USER return a string that contains the user ID of the current connection and can be used as a default value in columns with character data types. On UPDATE, columns with a default value of USER or CURRENT USER are not changed.

LAST USER Default

The special value LAST USER returns the name of the user who last modified the row and can be used as a default value in columns with character data types. On INSERT and LOAD, this constant has the same effect as CURRENT USER. On UPDATE, if a column with a default value of LAST USER is not explicitly modified, it is changed to the name of the current user.

When combined with the DEFAULT TIMESTAMP, a default value of LAST USER can be used to record (in separate columns) both the user and the date and time a row was last changed.

9.2.9 The IDENTITY or AUTOINCREMENT Default

The IDENTITY/AUTOINCREMENT default is useful for numeric data fields where the value of the number itself may have no meaning.

The feature assigns each new row a value of one greater than the current highest value in the column. You can use IDENTITY/AUTOINCREMENT columns to record purchase order numbers, to identify customer service calls or other entries where an identifying number is required.

Autoincrement columns are typically primary key columns or columns that are constrained to hold unique values (see CREATE TABLE statement in *SAP IQ SQL Reference*). For example, autoincrement default is effective when the column is the first column of an index, because the server uses an index or key definition to find the highest value.

You can sometimes retrieve the most recent value inserted into an autoincrement column using the <@@identity> global variable.

SAP IQ does not support the special value GLOBAL AUTOINCREMENT.

The initial IDENTITY/AUTOINCREMENT value is 0 when the table is created, and works only with positive integers.

A column with the AUTOINCREMENT default is referred to in Transact-SQL applications as an IDENTITY column. SAP IQ supports both keywords.

SAP IQ supports IDENTITY or DEFAULT AUTOINCREMENT as a default value. SAP IQ supports identity columns of any numeric type with any precision and scale 0, and the column can be NULL. SAP IQ identity columns must be positive and are limited by the range of the data type. SAP IQ supports a single identity column per table, and requires database option IDENTITY_INSERT set to a table name for explicit inserts and updates. To drop a table with an IDENTITY column, you cannot have IDENTITY_INSERT set to that table. The table can contain data when adding an identity column. Tables derived using SELECT INTO do not have Identity/Autoincrement columns. SAP IQ views cannot contain IDENTITY/DEFAULT AUTOINCREMENT columns.

9.2.10 The NEWID Default

Use UUIDs (Universally Unique IDentifiers), also known as GUIDs (Globally Unique IDentifiers) to uniquely identify rows in a table.

The values are generated such that a value produced on one computer does not match a value produced on another computer. UUIDs can therefore be used as keys in replication and synchronization environments.

See the `NEWID` function in *SAP IQ SQL Reference*.

9.2.11 The NULL Default

For columns that allow NULL values, specifying a NULL default is exactly the same as not specifying a default.

If the client inserting the row does not explicitly assign a value, the row automatically receives a NULL value.

You can use NULL defaults when information for some columns is optional or not always available.

See *NULL Value* in *SAP IQ SQL Reference*.

9.2.12 String and Number Defaults

You can specify a specific string or number as a default value, as long as the column holds a string or number data type.

You must ensure that the default specified can be converted to the data type of the column.

Default strings and numbers are useful when there is a typical entry for a given column. For example, if an organization has two offices, the headquarters in `city_1` and a small office in `city_2`, you may want to set a default entry for a location column to `city_1`, to make data entry easier.

9.2.13 Constant Expression Defaults

You can use a constant expression as a default value, as long as the expression does not reference database objects.

You can use functions such as `GETDATE` and `DATEADD` in a constant expression default value. If the default constant expression is not a function or simple value, you must enclose the expression in parentheses.

For example, constant expressions allow column defaults to contain entries such as the date fifteen days from today:

```
... DEFAULT ( DATEADD( DAY, 15, GETDATE() ) )
```

9.3 Table and Column Constraints

Constraints help to ensure that the data entered into a table is correct, and provide information to SAP IQ that boosts performance.

The `CREATE TABLE` statement and `ALTER TABLE` statement can specify many different attributes for a table. Along with the basic table structure (number, name and data type of columns, name and location of the table), you can specify other features that allow control over data integrity.

> ⚠ Caution
>
> Altering or creating tables may affect database users. For large tables, `ALTER TABLE` or `CREATE TABLE` can be time-consuming. `CREATE TABLE` processing delays execution of other processes until the statement completes. Although you can execute `ALTER TABLE` statements while other connections are active, no other requests that reference the altered table are allowed during statement execution.

In this section:

UNIQUE Constraints on Columns or Tables [page 195]
The `UNIQUE` constraint specifies that one or more columns uniquely identify each row in the table.

IQ UNIQUE Constraints on Columns [page 196]
`IQ UNIQUE` defines the expected cardinality of a column and determines whether the column loads as `Flat FP` or `NBit`.

CHECK Conditions on Columns [page 196]
Use a CHECK condition to specify a criterion for the values in a column.

CHECK Conditions on User-Defined Data Types [page 197]
You can attach CHECK conditions to user-defined data types. Columns defined on those data types inherit the CHECK conditions. A CHECK condition explicitly specified for a column overrides the CHECK condition from a user-defined data type.

CHECK Conditions as a Constraint on a Table [page 198]
You can apply a CHECK condition as a constraint on a table, instead of on a single column.

CHECK Conditions on Tables [page 198]
There are two ways to modify a CHECK condition defined on a table.

Removing Check Conditions from Tables [page 199]
Use the `ALTER TABLE` statement with the `DELETE CHECK` clause to remove all CHECK conditions on a table, including CHECK conditions on all its columns and CHECK conditions inherited from user-defined data types.

9.3.1 UNIQUE Constraints on Columns or Tables

The `UNIQUE` constraint specifies that one or more columns uniquely identify each row in the table.

`UNIQUE` is essentially the same as a `PRIMARY KEY` constraint, except that you can specify more than one `UNIQUE` constraint in a table. With both `UNIQUE` and `PRIMARY KEY`, columns cannot contain any `NULL` values.

Example 1

This example adds the column `ss_number` to the `employee` table, and ensures that each value in it is unique throughout the table:

```
ALTER TABLE employee
ADD ss_number char(11) UNIQUE
```

Example 2

In this example, three columns make a unique entry.

```
ALTER TABLE product
ADD UNIQUE (name, size, color)
```

9.3.2 IQ UNIQUE Constraints on Columns

`IQ UNIQUE` defines the expected cardinality of a column and determines whether the column loads as `Flat FP` or `NBit`.

`IQ UNIQUE` defines the expected cardinality of a column and determines whether the column loads as `Flat FP` or `NBit`. An `IQ UNIQUE(<n>)` value explicitly set to 0 loads the column as `Flat FP`. Columns without an IQ UNIQUE constraint implicitly load as `NBit` up to the limits defined by the `FP_NBIT_AUTOSIZE_LIMIT` and `FP_NBIT_LOOKUP_MB` options:

- `FP_NBIT_AUTOSIZE_LIMIT` limits the number of distinct values that load as `NBit`.
- `FP_NBIT_LOOKUP_MB` sets a threshold for the total `NBit` dictionary size.
- `FP_NBIT_ROLLOVER_MAX_MB` sets the dictionary size for implicit `NBit` rollovers from `NBit` to `Flat FP`.
- `FP_NBIT_ENFORCE_LIMITS` enforces `NBit` dictionary sizing limits. This option is OFF by default.

Using `IQ UNIQUE` with an `<n>` value less than the `FP_NBIT_AUTOSIZE_LIMIT` is not necessary. Auto-size functionality automatically sizes all low or medium cardinality columns as `NBit`. Use `IQ UNIQUE` in cases where you want to load the column as `Flat FP` or when you want to load a column as `NBit` when the number of distinct values exceeds the `FP_NBIT_AUTOSIZE_LIMIT`.

9.3.3 CHECK Conditions on Columns

Use a CHECK condition to specify a criterion for the values in a column.

The condition may specify rules that data must satisfy in order to be reasonable, or rules that reflect organization policies and procedures.

CHECK conditions on individual column values are useful when only a restricted range of values are valid for that column.

Example 1

The entry should match one of a limited number of values. To specify that a `city` column only contains one of a certain number of allowed cities (say, those cities where the organization has offices), you could use a constraint similar to:

```
ALTER TABLE office
MODIFY city
CHECK ( city IN ( 'city_1', 'city_2', 'city_3' ) )
```

By default, string comparisons are case insensitive unless the database is explicitly created as a case-sensitive database, using the `CASE RESPECT` option.

Example 2

A date or number falls in a particular range. You may want to require that the `start_date` column of an employee table must be between the date the organization was formed and the current date, as in:

```
ALTER TABLE employee
MODIFY start_date
CHECK ( start_date BETWEEN '1983/06/27'
                   AND CURRENT DATE )
```

You can use several date formats: the YYYY/MM/DD format is always recognized regardless of the current option settings.

9.3.4 CHECK Conditions on User-Defined Data Types

You can attach CHECK conditions to user-defined data types. Columns defined on those data types inherit the CHECK conditions. A CHECK condition explicitly specified for a column overrides the CHECK condition from a user-defined data type.

When defining a CHECK condition on a user-defined data type, any variable prefixed with the @ sign is replaced by the name of the column when the CHECK condition is evaluated. For example, the following user-defined data type accepts only positive integers:

```
CREATE DATATYPE posint INT
CHECK ( @col > 0 )
```

You can use any variable name prefixed with @ instead of `@col`. Any column defined using the `posint` data type accepts only positive integers unless it has a different explicitly specified CHECK condition.

An `ALTER TABLE` statement with the `DELETE CHECK` clause deletes all CHECK conditions from the table definition, including those inherited from user-defined data types.

For information on user-defined data types, see *SAP IQ SQL Reference*.

9.3.5 CHECK Conditions as a Constraint on a Table

You can apply a CHECK condition as a constraint on a table, instead of on a single column.

Such CHECK conditions typically specify that two values in a row being entered or modified have a proper relation to each other. Column CHECK conditions are held individually in the system tables, and can be replaced or deleted individually. CHECK conditions on individual columns are recommended where possible.

Action	ALTER TABLE Example

Add a new CHECK condition to a table or to an individual column	In this example, in a library database, the `date_returned` column for a particular entry must be the same as, or later than the `date_borrowed` entry:
	`ALTER TABLE loan`
	`ADD CHECK(date_returned >= date_borrowed)`

Delete a CHECK condition from an individual column	To delete a CHECK condition on a column, set it to NULL.
	This example removes the CHECK condition on the `phone` column in the `customer` table:
	`ALTER TABLE customer MODIFY phone`
	`CHECK NULL`

Replace a CHECK condition on a column	You can replace a CHECK condition on a column in the same way as you would add a CHECK condition.
	This example replaces the CHECK condition on the `city` column of the `office` table:
	`ALTER TABLE office`
	`MODIFY city`
	`CHECK ( city IN ( 'city_1', 'city_2', 'city_3' ) )`

9.3.6 CHECK Conditions on Tables

There are two ways to modify a CHECK condition defined on a table.

- Add a new CHECK condition using `ALTER TABLE` with an ADD table-constraint clause.
- Delete all existing CHECK conditions, including column CHECK conditions, using `ALTER TABLE` DELETE CHECK, and then add in new CHECK conditions.

Deleting a column from a table does not delete CHECK conditions associated with the column that are held in the table constraint. If the constraints are not removed, any attempt to query data in the table produces a `column not found` error message.

9.3.7 Removing Check Conditions from Tables

Use the `ALTER TABLE` statement with the `DELETE CHECK` clause to remove all CHECK conditions on a table, including CHECK conditions on all its columns and CHECK conditions inherited from user-defined data types.

Procedure

Issue the following statement:

```
ALTER TABLE <table_name>
DELETE CHECK
```

9.4 Entity and Referential Integrity

Once you specify the primary key for each table, no further action is needed by client application developers or the database administrator to maintain entity integrity.

The table owner defines the primary key for a table when creating it. If the structure of a table is later modified, the primary key can also be redefined using the `ALTER TABLE` statement clauses `DELETE PRIMARY KEY` or `ADD PRIMARY KEY`. See *SAP IQ SQL Reference*.

Some application development systems and database design tools allow you to create and alter database tables. If you are using such a system, you may not have to explicitly enter the `CREATE TABLE` or `ALTER TABLE` command. The application generates the statement from the information you provide.

When you insert or update a table row, the database server ensures that the primary key for the table is still valid: that each row in the table is uniquely identified by the primary key.

Example 1

The `Employees` table in the demo database uses an employee ID as the primary key. When a new employee is added to the table, IQ checks that the new employee ID value is unique, and is not NULL. See *Table Names and Owners* in *Introduction to SAP IQ* for a list of tables in the demo database.

Example 2

The `SalesOrderItems` table in the demo database uses two columns to define a primary key.

This table holds information about the items that can be ordered. One column contains an `id` that specifies an order, but there may be several items on each order, so this column by itself cannot be a primary key. An

additional `line_id` column identifies the line that corresponds to the item. Together, the columns `id` and `line_id` specify an item uniquely, and form the primary key. This is known as a multicolumn primary key.

In this section:

If a Client Application Breaches Entity Integrity [page 200]
Entity integrity requires that each value of a primary key or unique constraint be unique within the table, and that there are no NULL values.

Referential Integrity [page 200]
The entries in a foreign key must correspond to the primary key values of a row in the referenced table.

Concurrent Operations [page 205]
The referential integrity feature of SAP IQ restricts concurrent updates or deletes on a primary table during loads or inserts on a foreign table.

Disabling Referential Integrity Checking [page 207]
You can use the SAP IQ option `DISABLE_RI_CHECK` to bypass referential integrity checking.

9.4.1 If a Client Application Breaches Entity Integrity

Entity integrity requires that each value of a primary key or unique constraint be unique within the table, and that there are no NULL values.

If a client application attempts to insert or update a primary key value, and provides values that are not unique, entity integrity is breached.

A breach in entity integrity prevents the new information from being added to the database, and instead sends the client application an error.

The application programmer should decide how to present this information to the user and enable the user to take appropriate action. The appropriate action in this case is usually just to provide a unique value for the primary key.

SAP IQ checks referential integrity for each `UPDATE` on a foreign key or candidate key, each `DELETE` on a candidate key, and each `LOAD/INSERT` on a foreign key. When a referential integrity violation occurs, `UPDATE` or `DELETE` requests are immediately denied and rolled back. `LOAD/INSERT` requests that violate referential integrity are also denied or rolled back. SAP IQ also optionally rejects rows that violate data integrity as specified by the user.

9.4.2 Referential Integrity

The entries in a foreign key must correspond to the primary key values of a row in the referenced table.

Occasionally, some other unique column combination may be referenced instead of a primary key. The primary key or column (or set of columns) with a unique constraint is known as a `candidate key`. The referenced column or set of columns must be a candidate key and is called the `referenced key`.

In this section:

Loss of Referential Integrity [page 201]
SAP IQ provides protection against referential integrity loss.

Foreign Keys [page 201]
Use the `CREATE TABLE` statement or `ALTER TABLE` statement to create foreign keys, as you do primary keys.

Referential Integrity Violations [page 202]
SAP IQ supports referential integrity with RESTRICT action (the ANSI default) at the statement level.

Enforcing Referential Integrity with Existing Unenforced Foreign Keys [page 202]
You can enforce referential integrity with unenforced foreign keys.

Enforcing Referential Activity in a New Table [page 203]
Enforce referential integrity in a new table.

Dropping a Foreign Key [page 204]
Drop a foreign key from a table.

9.4.2.1 Loss of Referential Integrity

SAP IQ provides protection against referential integrity loss.

Your database can lose referential integrity if someone:

- Updates or deletes a primary key value that has a matching foreign key value. All the foreign keys referencing that primary key violate referential integrity.
- Adds a new row to the foreign table, and enters a value for the foreign key that has no corresponding candidate key value. The database violates referential integrity.

SAP IQ provides protection against both types of integrity loss.

When a referenced candidate key is updated or deleted, SAP IQ disallows `UPDATE` or `DELETE`.

9.4.2.2 Foreign Keys

Use the `CREATE TABLE` statement or `ALTER TABLE` statement to create foreign keys, as you do primary keys.

> **i Note**
>
> You cannot create foreign key constraints on local temporary tables. Global temporary tables must be created with `ON COMMIT PRESERVE ROWS`.

The demo database contains an employee table and a department table. The primary key for the employee table is the employee ID, and the primary key for the department table is the department ID.

For example, assume the following schema:

```
DEPT table
  ( DeptNo int primary key
  DeptName varchar(20),
  Mgr int,
```

```
  foreign key MGR_EMPNO (Mgr) references EMPLOYEE(EmpNo) on update restrict }
EMPLOYEE table
  { EmpNo int primary key,
  DeptNo int references DEPT(DeptNo) on delete restrict,
  LastName varchar(20),
  FirstName varchar(20),
  Salary int }
```

In the employee table, the department ID is a foreign key for the department table; each department ID in the employee table corresponds exactly to a department ID in the department table.

The foreign key relationship is a many-to-one relationship, and, in this example, is mandatory. Several entries in the employee table have the same department ID entry, but the department ID is the primary key for the department table, and so is unique. If a foreign key was allowed to reference a column with duplicate entries in the department table, there is no way to determine which row in the department table is the appropriate reference. This is a mandatory foreign key.

9.4.2.3 Referential Integrity Violations

SAP IQ supports referential integrity with RESTRICT action (the ANSI default) at the statement level.

This means that SAP IQ denies requests for updates and deletes on a primary key or any column with a unique constraint that removes any value upon which correspondent foreign keys depend. (You must carefully consider the order in which you request deletes and updates.) SAP IQ issues an error message and rolls back load operations that violate referential integrity, but does allow you to specify that certain rows can be ignored.

9.4.2.4 Enforcing Referential Integrity with Existing Unenforced Foreign Keys

You can enforce referential integrity with unenforced foreign keys.

Context

Assume the following schema:

```
DEPT table
  { DeptNo int primary key
  DeptName varchar(20),
  Mgr int,
  foreign key MGR_EMPNO (Mgr) references EMPLOYEE(EmpNo) on update restrict }
EMPLOYEE table
  { EmpNo int primary key,
  DeptNo int references DEPT(DeptNo) on delete restrict,
  LastName varchar(20),
  FirstName varchar(20),
  Salary int }
```

Procedure

1. Identify the candidate key to foreign key relationship.

   In the schema, there are two such relationships:
   - Foreign key (`EMPLOYEE.DeptNo`) to Candidate key (`DEPT.DeptNo`)
   - Foreign key (`DEPT.Mgr`) to Candidate key (`EMPLOYEE.EMPNo`)
2. Add a primary key or unique constraint on the candidate key via the `ALTER TABLE` statement if none exist. (In the preceding example, the primary key already exists.) All candidate key values must be unique and non-null.
3. Drop any existing unenforced foreign key constraint.

   For example:

   ```
   ALTER TABLE DEPT DROP FOREIGN KEY MGR_EMPNO;
   ALTER TABLE EMPLOYEE DROP FOREIGN KEY DEPT;
   ```

   In the schema, the unenforced foreign key constraints `MGR_EMPNO` and `EMPLOYEE(DeptNo)` referencing `DEPT(DeptNo)` must be dropped. If there is no user specified role name for `EMPLOYEE(DeptNo)` to `DEPT(DeptNo)`, the default role name is the same as the primary table; in other words, DEPT.
4. Add the foreign key constraint(s).

   For example:

   ```
   ALTER TABLE DEPT ADD FOREIGN KEY MGR_EMPNO(Mgr) REFERENCES EMPLOYEE(EmpNo);
   ALTER TABLE EMPLOYEE ADD FOREIGN KEY EMP_DEPT(DeptNo) REFERENCES DEPT(DeptNo);
   ```

9.4.2.5 Enforcing Referential Activity in a New Table

Enforce referential integrity in a new table.

Procedure

1. Create the primary table, for example:

   ```
   CREATE TABLE DEPT(DeptNo int primary key,
     DeptName varchar(20),
     Mgr int );
   ```
2. Create the foreign table. In this example, the default role name for the specified foreign key is DEPT:

   ```
   CREATE TABLE EMPLOYEE(EmpNo int primary key,
     DeptNo int REFERENCES DEPT(DeptNo)
     ON DELETE RESTRICT,
     LastName varchar(20),
     FirstName varchar(20),
     Salary int);
   ```

3. Specify the foreign key constraint. In this example, the user specified role name for the same foreign key is EMP_DEPT.

```
ALTER TABLE DEPT ADD FOREIGN KEY EMP_DEPT(DeptNo)
  REFERENCES DEPT(DeptNo);
```

9.4.2.6 Dropping a Foreign Key

Drop a foreign key from a table.

Procedure

1. Create the primary table and foreign table. For example:

```
CREATE TABLE DEPT(DeptNo int primary key,
DeptName varchar(20),
Mgr int );
```

```
CREATE TABLE EMPLOYEE(EmpNo int primary key,
DeptNo int references DEPT(DeptNo)
on delete restrict,
LastName varchar(20),
FirstName varchar(20),
Salary int);
```

2. When there is no role name assigned, the default role name for the specified foreign key is DEPT:

```
ALTER TABLE EMPLOYEE DROP FOREIGN KEY DEPT;
```

3. If there are multiple foreign keys and the role name is unknown, you can use the `sp_iqconstraint` procedure to display it. See *SAP IQ SQL Reference*.
4. If a role name was assigned, EMP_DEPT, for example, you must specify it when dropping the key:

```
ALTER TABLE EMPLOYEE DROP FOREIGN KEY EMP_DEPT;
```

Example

These statements do not drop the non-unique HG index for EMPLOYEE(DeptNo), which is implicitly created. To drop it, use `sp_iqindex` to find the High_Group index name and use the `DROP INDEX` statement, as follows:

```
sp_iqindex('EMPLOYEE');
```

```
EMPLOYEE DBA DeptNO FP ASIQ_IDX_T27_C2_FP N
EMPLOYEE DBA DeptNO HG ASIQ_IDX_T27_C2_HG N
EMPLOYEE DBA EmpNO FP ASIQ_IDX_T27_C1_FP N
EMPLOYEE DBA EmpNO HG ASIQ_IDX_T27_I11_HG N
EMPLOYEE DBA FirstName FP ASIQ_IDX_T27_C4_FP N
```

```
EMPLOYEE DBA LastName FP ASIQ_IDX_T27_C3_FP N
EMPLOYEE DBA Salary FP ASIQ_IDX_T27_C5_FP N
```

```
DROP INDEX ASIQ_IDX_T27_C2_HG
```

To drop a table, you must drop all associated foreign key constraints. Drop foreign key constraint and tables in this order:

```
ALTER TABLE DROP FOREIGN KEY MGR_EMPNO;
DROP TABLE EMPLOYEE;
DROP TABLE DEPT;
```

Another way to drop the same tables would be to use the following two `ALTER TABLE` statements in any order and then do `DROP TABLE` statements in any order:

```
ALTER TABLE DEPT DROP FOREIGN KEY MGR_EMPNO;
ALTER TABLE EMPLOYEE DROP FOREIGN KEY EMP_DEPT;
```

Suppose that the database also contained an office table, listing office locations. The employee table might have a foreign key for the office table that indicates where the employee's office is located. The database designer may allow for an office location not being assigned when the employee is hired. In this case, the foreign key should allow the NULL value for when the office location is unknown or when the employee does not work out of an office.

9.4.3 Concurrent Operations

The referential integrity feature of SAP IQ restricts concurrent updates or deletes on a primary table during loads or inserts on a foreign table.

Concurrent operations that return an error:

First Request	Request of Overlapping Transaction
Request by one transaction for `LOAD/INSERT/UPDATE/ALTER TABLE ADD` foreign key/ `ALTER TABLE DROP` foreign key to any foreign table	`DELETE` its associated primary table with deletable row(s).
`UPDATE` its associated primary table.	
`TRUNCATE` its associated primary table.	

SAP IQ also generates an error for a request by one transaction to alter a table to add a foreign key or drop a foreign key while there are old versions of the foreign table, the primary table, or both, in use by other transactions.

For both enforced and unenforced foreign key and primary key, SAP IQ allows:

- Simultaneous load or insert on one or more foreign tables and the shared primary table.
- Simultaneous load or insert on foreign tables and delete, or update, or truncate on another one or more foreign tables.
- Simultaneous delete, or update, or truncate on two or more foreign tables, even if sharing the same primary table.
- Simultaneous delete, or update, or truncate on foreign tables and delete, or update, or truncate on the shared primary table.

- `ALTER TABLE ADD` foreign key or `DROP` foreign key if no transaction is using any old version(s) of foreign/primary table and these unused old version(s) will be dropped as part of the `ADD/DROP` foreign key operation.

Concurrent Operations on Foreign and Primary Tables

The table-level versioning of SAP IQ guarantees consistent referential integrity checks while allowing concurrent load, or insert, or update operations on the foreign table and `LOAD/INSERT` operations on the primary table.

SAP IQ also verifies that there are no deleted old values in a foreign table when a transaction requesting `DELETE` or `UPDATE` starts. This provides consistent referential integrity checking during a concurrent delete operation on a foreign table and a delete or update operation on a PRIMARY Table.

Assume that there are two foreign key constraints among two foreign tables, `ftab1` and `ftab2`, and one primary table, `ptab`. Assume that foreign key `ftab1(fk1,fk2)` references candidate key `ptab(pk1,pk2)`. Foreign key `ftab2(fk1,fk2)` references the same candidate key. Candidate key `ptab(pk1,pk2)` can either be a primary key or a unique constraint.

This table shows which operations on both tables (foreign and primary) should be allowed and which return an error. Information in the table applies only to enforced foreign keys and candidate key.

	LOAD or INSERT ftab1	DELETE/ TRUNCATE TABLE ftab1	UPDATE ftab1 (fk1,fk2)	Populate New Index Non-FK ftab1 (fk1,fk2)	ADD FK ftab1 (fk1 fk2)	DROP FK ftab1 (fk2, fk2)
LOAD ftab2	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
LOAD ptab	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
INSERT ftab2	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
INSERT ptab	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
DELETE ftab2 TRUNCATE TABLE ftab2	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
DELETE ptab TRUNCATE TABLE ptab	Error	Allowed	Error	Allowed	Error	Error
UPDATE ftab2(fk1,fk2)	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
UPDATE ptab (pk1,pk2)	Error	Allowed	Error	Allowed	Error	Error
Populate new index	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
QUERY (old version of ftab1/ptab in use with or without (fk1,fk2))	Allowed	Allowed	Allowed	Allowed	Error	Error

	LOAD or INSERT ftab1	DELETE/ TRUNCATE TABLE ftab1	UPDATE ftab1 (fk1,fk2)	Populate New Index Non-FK ftab1 (fk1,fk2)	ADD FK ftab1 (fk1 fk2)	DROP FK ftab1 (fk2, fk2)
No old version of ftab2 in use	N/A	N/A	N/A	N/A	Allowed (drop all unused old versions of ftab1)	Allowed (drop all unused old versions of ftab1)

Concurrency conflict occurs if one transaction loads foreign key columns while another updates associated candidate key columns. There is no conflict if one transaction loads foreign key columns while another updates unassociated candidate key columns on one of its associated candidate tables.

> **i** Note
>
> For efficient performance, a query on union all views opens the tables referred to by those columns used as join keys or group by columns. Until the transaction commits and read locks on the tables are released, you cannot alter or drop the tables that have foreign keys used as join conditions or grouping columns. You can, however, load, insert, delete, and update these tables while the query is running.

9.4.4 Disabling Referential Integrity Checking

You can use the SAP IQ option `DISABLE_RI_CHECK` to bypass referential integrity checking.

> **i** Note
>
> Use this option carefully; bypassing referential integrity checking defeats the purpose of having the feature.

9.5 Integrity Rules in System Tables

All the information about integrity checks and rules in a database is held in system tables and views.

System Table	Description
`SYS.SYSTABLE`	CHECK constraints are held in the `view_def` column of `SYS.SYSTABLE`. For views, the `view_def` holds the `CREATE VIEW` command that created the view. You can check whether a particular table is a base table or a view by looking at the `table_type` column, which is BASE or VIEW.
`SYS.SYSFOREIGNKEYS`	This view presents the foreign key information from the two tables `SYS.SYSFOREIGNKEY` and `SYS.SYSFKCOL` in a more readable format.

System Table	Description
SYS.SYSCOLUMNS	This view presents the information from the SYS.SYSCOLUMN table in a more readable format. It includes default settings and primary key information for columns.

You can use Interactive SQL to browse these tables and views.

9.6 Validating Catalog Store Indexes

You can validate an index on SAP SQL Anywhere tables in the catalog store to ensure that every row referenced in the index actually works in the table.

Procedure

To validate an index, open a command prompt and run the `dbvalid` utility.

For example, the following statement, which should be typed on a single line) validates an index called EmployeeIndex. The `-i` switch specifies that each object name given is an index.

```
dbvalid -c "uid=dba;pwd=MyIQD@tabase;eng=myserver"
-i EmployeeIndex
```

See *dbvalid Database Administration Utility* in the *SAP IQ Utility Reference*.

10 Manage Transactions and Versioning

Transaction processing ensures that logically related groups commands execute as a unit. SAP IQ uses transaction processing to allow many users to read from the database while it is being updated. Transactions are fundamental to maintaining the accuracy of your data, and to data recovery in the event of system failure.

A crucial aspect of transaction processing is its ability to isolate users from the effect of other users' transactions. The SAP IQ approach to transaction processing, called snapshot versioning, supports the highest level of isolation recognized by ISO. Snapshot versioning provides each database user with a snapshot of the database. Any changes that you make to the database are not seen by other database users until you commit your transaction.

In this section:

Transactions [page 210]
: Each transaction is a sequence of logically related commands that accomplish one task and transform the database from one consistent state into another.

Transaction Blocking [page 212]
: When you set the `BLOCKING` option to on, any transaction attempting to obtain a write lock that conflicts with an existing write lock held by another transaction waits until the conflicting lock is released, or until the `BLOCKING_TIMEOUT` threshold is reached. By default, `BLOCKING` is off.

Versions [page 217]
: The database server binds the snapshot version when a transaction is created, after the transaction has obtained the necessary write locks on the table or the table row. Table-level versioning enables single-writer access and table-level locking. Row-level versioning (RLV) enables concurrent writer access and row-level locking for RLV-enabled tables.

Investigating Lock Contention Effects on Performance [page 233]
: Some load or query performance issues result from lock contention. If your kernel system time is greater than 10%, you may be experiencing lock contention.

Checkpoints, Savepoints, and Transaction Rollback [page 235]
: Besides permitting concurrency, snapshot versioning transaction processing plays an important role in data recovery.

Cursors in Transactions [page 240]
: Return the results of a `SELECT` in the form of a data type called a cursor.

Remote Transactions [page 243]
: Transaction management involving remote servers uses a two-phase commit protocol.

Related Information

Table-Level Snapshot Versioning [page 218]
Row-Level Snapshot Versioning [page 224]

10.1 Transactions

Each transaction is a sequence of logically related commands that accomplish one task and transform the database from one consistent state into another.

Transactions are atomic; SAP IQ executes all the statements within a transaction as a unit. At the end of each transaction, changes are committed to make them permanent. If for any reason any of the commands in the transaction do not process properly, some or all of the intermediate changes can be undone, or rolled back. The user application controls the conditions under which changes are committed or rolled back.

Transactions are made up of small blocks. The completion of each block marks a point at which the information is self-consistent. Transaction processing is fundamental to ensuring that a database contains correct information.

In most cases, SAP IQ transactions begin and end automatically, based on the commands being issued, and the options set. You can also issue explicit commands to begin or end a transaction.

Transactions start automatically with the first statement following either:

- A connection to a database
- The end of a previous transaction

SAP IQ also supports SAP IQ Transact-SQL commands, such as `BEGIN TRANSACTION`, for compatibility with SAP Adaptive Server Enterprise. You can explicitly start a transaction using the `BEGIN TRANSACTION` command.

Transactions complete with one of these events:

- A `COMMIT` statement makes the changes to the database permanent.
- A `ROLLBACK` statement undoes all the changes made by the transaction.
- A disconnection from a database causes an implicit rollback (the default) or commit.
- The execution of a statement with a side effect of an automatic commit makes changes to the database.

Database definition commands, such as `ALTER`, `CREATE`, and `DROP` all have the side effect of an automatic commit.

You can use savepoints to identify important states within a transaction and return to them selectively or cause other actions to occur.

In this section:

Viewing Transaction Activity [page 211]
: Use the `sp_iqtransaction` stored procedure to display a snapshot of transaction activity, such as main and temporary space created and in use, open cursors, and savepoints.

Isolation Levels [page 211]
: An important aspect of transaction processing is the ability to isolate modifications made to the database by concurrently executing transactions. In a database system, concurrently executing applications can encounter various forms of update anomalies.

10.1.1 Viewing Transaction Activity

Use the `sp_iqtransaction` stored procedure to display a snapshot of transaction activity, such as main and temporary space created and in use, open cursors, and savepoints.

Procedure

Run `sp_iqtransaction`.

See *sp_iqtransaction Procedure* in SAP IQ SQL Reference for complete syntax information.

The procedure returns a row for each transaction control block in the IQ transaction manager.

10.1.2 Isolation Levels

An important aspect of transaction processing is the ability to isolate modifications made to the database by concurrently executing transactions. In a database system, concurrently executing applications can encounter various forms of update anomalies.

Update anomalies include:

Dirty reads Transaction A modifies an object, but doesn't commit or roll back the change. Transaction B reads the modified object. Then transaction A further changes the object before performing a COMMIT or ROLLBACK. In this situation, transaction B has seen the object in a state that was never committed.

Nonrepeatable reads Transaction A reads an object. Transaction B then modifies or deletes the object and performs a COMMIT. If transaction A attempts to read the same object again, it has been changed or deleted.

Phantom data elements Transaction A reads a set of data that satisfies some condition. Transaction B then executes an INSERT and then a COMMIT. The newly committed data satisfies the condition for the statement issued by Transaction A previously. Transaction A then repeats the initial read and obtains a different set of data.

Lost updates In an application that uses cursors, Transaction A writes a change for a set of data. Transaction B then saves an update that is based on earlier data. The changes made by Transaction A are lost.

The function of concurrency control mechanisms in a DBMS is to permit the user to control the types of anomalies that their application can tolerate, if any. In many database systems, there's a trade-off between database consistency and performance, since many systems implement concurrency control through an often-complex set of locking mechanisms. The ISO SQL Standard defines four levels of isolation (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, and SERIALIZABLE) that provide a greater degree of protection from the anomalies listed earlier.

In SAP IQ, the concurrency control mechanism used is termed Table-Level Versioning (TLV), which is a form of snapshot isolation. With snapshot isolation, each concurrent transaction "sees" a consistent version of the database, including its own updates, but is insulated from seeing the updates performed by other concurrent transactions.

In SAP IQ, a transaction begins with its first SELECT or update DML statement, and from that point onwards, the application "sees" a consistent state of the database until it performs a COMMIT or ROLLBACK. A subsequent SELECT or DML statement triggers a subsequent transaction, and hence another snapshot.

TLV maintains this high level of isolation between concurrent read and write transactions by following these rules:

- The transaction manager maintains a snapshot of committed data when each transaction begins.
- Only one transaction can modify a particular table at any one time.
- A read operation on a table is never blocked by a write transaction.
- A transaction's writes are reflected in the snapshot it sees.
- Once a transaction begins, updates made by other transactions are invisible to it.

The level of isolation that TLV provides prevents most of the types of anomalies – dirty reads, nonrepeatable reads, and phantom rows – from occurring, although TLV doesn't guarantee transaction serializability. Serializability is the notion of a series of concurrent reads and updates of concurrently executing transactions that are semantically equivalent to some serial execution of those same transactions.

In summary, TLV:

1. Ensures that only one user can modify a table at any given time,
2. Keeps the changes invisible to other users until the changes are complete,
3. Provides a consistent view of the database for the duration of each transaction.

10.2 Transaction Blocking

When you set the `BLOCKING` option to on, any transaction attempting to obtain a write lock that conflicts with an existing write lock held by another transaction waits until the conflicting lock is released, or until the `BLOCKING_TIMEOUT` threshold is reached. By default, `BLOCKING` is off.

Connection blocking puts the requesting transaction to sleep (blocking the table-level or row-level lock) until the connection holding the lock releases it (unblocking the table-level or row-level lock). You control the duration of the block by setting the `BLOCKING_TIMEOUT` value in milliseconds. When `BLOCKING_TIMEOUT` is 0 (the default), all blocked transactions in the connection wait indefinitely until the connection obtains the requested lock. When `BLOCKING_TIMEOUT` is set, and the lock is not released within the specified time, then the waiting transaction receives an error message.

Blocking takes advantage of delayed transaction versioning, where the transaction manager creates the transaction snapshot version after establishing a table-level or row-level lock.

Consider this blocking example for a table-level lock:

- User A connects.
- User A executes **`INSERT INTO iq_table1 VALUES( 3, 300);`**.
- User B connects.
- User B executes **`INSERT INTO iq_table1 VALUES( 4, 400);`**.

With connection blocking disabled, User B's transaction is rolled back, and he or she receives an error indicating that another user has a lock on `iq_table1`.

With connection blocking enabled (and `BLOCKING_TIMEOUT` set to 0), User B's `INSERT` statement is put to sleep until User A's `INSERT` statement commits, releasing the table write lock.

With connection blocking enabled and `BLOCKING_TIMEOUT` set to 200 milliseconds, User B's transaction is rolled back and he or she receives an error indicating that another user has a lock on `iq_table1`, if User A's transaction does not commit within the 200 millisecond blocking timeout threshold.

For row-level lock examples, see *SAP IQ Administration: In-Memory Row-Level Versioning > Manage Blocking in RLV Store*.

In this section:

Enabling Connection Blocking [page 213]
: Enable connection blocking to force any transaction attempting to obtain a lock that conflicts with another transaction's existing lock to wait: either until every conflicting lock is released, or until the `BLOCKING_TIMEOUT` threshold is reached.

Disabling Connection Blocking [page 214]
: Disable connection blocking to force any transaction attempting to obtain a lock that conflicts with another transaction's existing lock to roll back the transaction and display an error.

Setting the Blocking Timeout Threshold [page 214]
: Use the threshold to set the length of time, in milliseconds, a transaction waits to obtain a lock. If the transaction attempting to obtain a lock conflicts with another transaction's existing lock, it waits until the `BLOCKING_TIMEOUT` option threshold is reached. If the conflict still exists, the transaction rolls back and you see an error.

Transaction Blocking Deadlocks [page 215]
: Transaction blocking can lead to a deadlock situation, in which a set of transactions arrive at a state where none of them can proceed.

10.2.1 Enabling Connection Blocking

Enable connection blocking to force any transaction attempting to obtain a lock that conflicts with another transaction's existing lock to wait: either until every conflicting lock is released, or until the `BLOCKING_TIMEOUT` threshold is reached.

Prerequisites

Requires SET ANY PUBLIC OPTION system privilege.

Context

The blocking option can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Procedure

Enable the `BLOCKING` database option by executing the following:

```
SET TEMPORARY OPTION BLOCKING = 'ON'
```

See *BLOCKING Option* in SAP IQ SQL Reference for complete syntax information.

10.2.2 Disabling Connection Blocking

Disable connection blocking to force any transaction attempting to obtain a lock that conflicts with another transaction's existing lock to roll back the transaction and display an error.

Prerequisites

Requires SET ANY PUBLIC OPTION system privilege.

Context

The blocking option can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Procedure

Disable the `BLOCKING` database option by executing the following:

```
SET TEMPORARY OPTION BLOCKING = 'OFF'
```

See *BLOCKING Option* in SAP IQ SQL Reference for complete syntax information.

10.2.3 Setting the Blocking Timeout Threshold

Use the threshold to set the length of time, in milliseconds, a transaction waits to obtain a lock. If the transaction attempting to obtain a lock conflicts with another transaction's existing lock, it waits until the `BLOCKING_TIMEOUT` option threshold is reached. If the conflict still exists, the transaction rolls back and you see an error.

Prerequisites

Requires SET ANY PUBLIC OPTION system privilege.

Context

The default value, 0, indicates that a blocked transaction must wait indefinitely until all conflicting transactions release their locks.

Procedure

To set the `BLOCKING_TIMEOUT` database option value to the number of milliseconds you want the transaction to wait for conflicting transactions to release their locks, execute:

```
SET TEMPORARY BLOCKING_TIMEOUT = '<timeout-value>'
```

See *BLOCKING_TIMEOUT Option* in SAP IQ SQL Reference for complete syntax information.

This statement sets the blocking timeout value to 400 milliseconds:

```
SET TEMPORARY BLOCKING_TIMEOUT = '400'
```

10.2.4 Transaction Blocking Deadlocks

Transaction blocking can lead to a deadlock situation, in which a set of transactions arrive at a state where none of them can proceed.

A deadlock can arise for two reasons:

- Cyclical blocking conflict – transaction A is blocked on transaction B, and transaction B is blocked on transaction A. Additional time cannot solve the problem, and one of the transactions must be canceled, allowing the other to proceed. The same situation can arise with more than two transactions blocked in a cycle.
  To eliminate a transactional deadlock, the database server selects a connection from those involved in the deadlock, rolls back the changes for the transaction that is active on that connection and returns an error. The database server selects the connection to roll back by using an internal heuristic that prefers the connection with the smallest blocking wait time left as determined by the `BLOCKING_TIMEOUT` option. If all connections are set to wait forever, then the connection that caused the server to detect a deadlock is selected as the victim connection.
- All workers are blocked – when a transaction becomes blocked, its worker is not relinquished. For example, a database server is configured with three workers. Transactions A, B, and C are blocked on transaction D, which is not currently executing a request. A deadlock situation arises because there are no available workers. This situation is called thread deadlock.

Suppose that the database server has <n> workers. Thread deadlock occurs when <n>-1 workers are blocked, and the last worker is about to block. The database server's kernel cannot permit this last worker to block, since doing so results in all workers being blocked, and the database server stops responding. Instead, the database server ends the task that is about to block the last worker, rolls back the changes for the transaction active on that connection, and returns an error.
Database servers with tens or hundreds of connections may experience thread deadlocks in long-running requests, either because of the size of the database or because of blocking. In this case, you may want to increase the value of the `-gn` server option of the `start_iq` utility.

You can monitor locks and deadlocks using SAP IQ Cockpit. For more information, see *Monitor SAP IQ* in the *SAP IQ Cockpit* documentation.

In this section:

Creating a Deadlock Reporting Event in Interactive SQL [page 216]
Create a table and a system event for obtaining information about deadlocks.

10.2.4.1 Creating a Deadlock Reporting Event in Interactive SQL

Create a table and a system event for obtaining information about deadlocks.

Prerequisites

- You are connected as the DBA user. This ensures you have the required privileges to complete each task in the tutorial.
- RLV storage is configured.
- The tables referenced do not already exist, or if they do, do not contain any unmerged RLV transactions, as this may result in different output throughout the tutorial.

Context

See *sa_report_deadlocks System Procedure*, *sa_server_option System Procedure*, and *LOG_DEADLOCKS Option* in SAP IQ SQL Reference for complete syntax information.

Procedure

1. Create a table to store the data returned from the `sa_report_deadlocks` system procedure:

```
CREATE TABLE DeadlockDetails(
  deadlockId INT PRIMARY KEY DEFAULT AUTOINCREMENT,
```

```
  snapshotId BIGINT,
  snapshotAt TIMESTAMP,
  waiter INTEGER,
  who VARCHAR(128),
  what LONG VARCHAR,
  object_id UNSIGNED BIGINT,
  record_id BIGINT,
  owner INTEGER,
  is_victim BIT,
  rollback_operation_count UNSIGNED INTEGER );
```

2. Create an event that sends an e-mail notification when a deadlock occurs:

```
CREATE EVENT DeadlockNotification
TYPE Deadlock
HANDLER
BEGIN
 INSERT INTO DeadlockDetails WITH AUTO NAME
 SELECT snapshotId, snapshotAt, waiter, who, what, object_id, record_id,
        owner, is_victim, rollback_operation_count
    FROM sa_report_deadlocks ();
 COMMIT;
 CALL xp_startmail ( mail_user ='John Smith',
                     mail_password ='mypwd' );
 CALL xp_sendmail( recipient='DBAdmin',
                   subject='Deadlock details added to the DeadlockDetails
table.' );
 CALL xp_stopmail ( );
END;
```

   This event copies the results of the `sa_report_deadlocks` system procedure into a table and notifies the administrator about the deadlock.

3. Enable the `LOG_DEADLOCKS` option:

```
SET OPTION PUBLIC.log_deadlocks = 'On';
```

4. Enable logging of the most-recently executed statement:

```
CALL sa_server_option( 'RememberLastStatement', 'YES' );
```

10.3 Versions

The database server binds the snapshot version when a transaction is created, after the transaction has obtained the necessary write locks on the table or the table row. Table-level versioning enables single-writer access and table-level locking. Row-level versioning (RLV) enables concurrent writer access and row-level locking for RLV-enabled tables.

In this section:

Table-Level Snapshot Versioning [page 218]
The default transaction processing behavior, table-level snapshot versioning, allows multiple writers to modify a table serially—that is, one after the other, never more than one at a time—while multiple readers continue to work on an original copy of the table.

Row-Level Snapshot Versioning [page 224]

Row-level snapshot versioning applies only to tables enabled for in-memory RLV storage. Row-level snapshot versioning allows multiple writers to make concurrent DML changes to a table, but never to the same rows at the same time.

Viewing the Versioning Type of a Transaction [page 232]
Use the `sp_iqtransaction` stored procedure to view the versioning type (either table-level or row-level) of each active transaction.

Temporary Table Versioning [page 233]
A temporary table that is created in the database is called a global temporary table. A temporary table that is declared (rather than created in the database) is called a local temporary table.

10.3.1 Table-Level Snapshot Versioning

The default transaction processing behavior, table-level snapshot versioning, allows multiple writers to modify a table serially—that is, one after the other, never more than one at a time—while multiple readers continue to work on an original copy of the table.

Without table-level snapshot versioning, concurrent read and write operations might cause inconsistencies in the IQ main store. The table-level versioning provided by SAP IQ prevents inconsistencies both by serializing transactions, and by making the table the version level.

While any transaction processing system is designed to ensure that the database remains consistent, the SAP IQ approach means that users don't need to worry about placing their queries and updates in appropriate transactions. SAP IQ begins and ends transactions automatically, and ensures that read and write operations do not interfere with each other.

When table-level snapshot versioning is enabled but the table is configured for RLV storage, concurrent read and write operations are still disallowed. The table-level snapshot versioning setting overrides a table's RLV setting.

In this section:

Enabling Table-Level Snapshot Versioning [page 219]
Table-level snapshot versioning is enabled by default.

Table-Level Locking [page 219]
All table-level locks occur automatically, based on the type of operation a user requests.

Displaying Active Table Locks [page 222]
Identify the user who has a table locked. An attempt to write to a table fails if another transaction holds a lock on that table.

Performance Implications of Table-Level Snapshot Versioning [page 222]
Typically, table-level snapshot versioning has a minimal impact on performance.

10.3.1.1 Enabling Table-Level Snapshot Versioning

Table-level snapshot versioning is enabled by default.

Procedure

- If the default versioning method has been changed, use the `ALLOW_SNAPSHOT_VERSIONING` database option to enable it.
- To temporarily set table-level versioning, issue the following:

```
SET TEMPORARY OPTION Snapshot_Versioning = 'Table-level'
```

10.3.1.2 Table-Level Locking

All table-level locks occur automatically, based on the type of operation a user requests.

You need not explicitly request a lock; the transaction that has access to the table holds the lock.

When a table is locked in SAP IQ, no other transaction, except data definition operations, can have write access to it. Any other write transaction that attempts to access a table with a write lock on it receives an error. Any transaction can obtain read access.

The locks maintain the reliability of information in the database by preventing concurrent access by other transactions. The database server retains all the locks acquired by a transaction until the transaction completes, due to either a commit or a rollback.

You can reserve write locks on a set of tables within a new transaction using the `LOCK TABLE` statement. `LOCK TABLE` commits the current transaction and allows transactions to enqueue until the locks are available.

In this section:

Table Locks for DML Operations [page 220]
Data manipulation language (DML) operations include insertions, deletions, updates, and queries. For all such operations, table-level snapshot versioning permits one writer and multiple readers on any given table.

Table Locks for DDL Operations [page 220]
Data definition language (DDL) operations include `CREATE`, `DROP`, and `ALTER`.

Primary Keys and Locking [page 222]
Because only one user can update a table when table-level snapshot versioning is enabled, primary key generation does not cause concurrency conflicts.

10.3.1.2.1 Table Locks for DML Operations

Data manipulation language (DML) operations include insertions, deletions, updates, and queries. For all such operations, table-level snapshot versioning permits one writer and multiple readers on any given table.

In table-level snapshot versioning:

- Read transactions do not block write transactions.
- Write transactions do not block read transactions.
- A single update user and multiple read-only users can concurrently access a table.
- Only one user can update the data in a given table at one time.

The first transaction to open a table in write mode gains access to the table. A second transaction that tries to open the table in write mode receives an error. Any additional attempts to write to the table in the current transaction fail. The transaction can continue, but only with read operations or with writes to other tables.

SAP IQ supports share, write, and exclusive lock enqueuing, allowing you to lock a table for a specified period. You can write lock multiple tables simultaneously.

To avoid future version errors from subsequent DML statements, use the `LOCK TABLE` statement to reserve a write lock on the table or set of tables that you plan to modify.

In the case of deadlocks, the last `LOCK TABLE` statement that became blocked is usually rolled back and an error returns to that transaction about the form of deadlock that occurred.

In certain cases, you must issue `COMMIT` or `ROLLBACK` statements. For example, an explicit `COMMIT` or `ROLLBACK` is required to release locks when DML statements fail due to integrity constraints.

If a DML statement fails due to a referential integrity violation on the referenced table or an unavailable lock on other tables, you see: SQL Anywhere Error -210.

10.3.1.2.2 Table Locks for DDL Operations

Data definition language (DDL) operations include `CREATE`, `DROP`, and `ALTER`.

DDL operations on a given table or index lock out all other readers and writers from any table being modified. This approach is crucial to the accuracy of query results. It ensures, for example, that a table column does not disappear from the database while you are selecting data from that column.

`CREATE`, `DROP`, and `ALTER` commands have special properties:

- They cannot start while any other transaction is using the table or index they are modifying.
  For example, if a user issues a SELECT on a table, the table is locked and cannot be altered until the user logs out, issues a SELECT on another table, or issues a ROLLBACK.
- They include an automatic `COMMIT` on completion.
- Existing transactions that try to use the table being modified receive an error. In other words, if you are accessing a table, and a DDL command changes that table, your command fails.
- At any given time, only one of the commands `CREATE DBSPACE`, `DROP DBSPACE`, and `CHECKPOINT` can be executing in a database.

In this section:

DDL Locking Errors [page 221]
Errors may occur if you issue a DDL command when another DDL command is in process.

10.3.1.2.2.1 DDL Locking Errors

Errors may occur if you issue a DDL command when another DDL command is in process.

You may see this error message:

```
Cannot perform DDL command now on table <tablename>
 as a DDL command is already in progress on that table.
```

If a `CREATE DBSPACE` or `DROP DBSPACE` command is in progress, and a user explicitly issues a `CHECKPOINT` command, the checkpoint fails with:

```
Run time SQL Error
```

If a `CHECKPOINT` command is in progress, a user who issues a `CREATE DBSPACE` or `DROP DBSPACE` command sees:

```
Cannot perform requested command as there is a
CHECKPOINT command in progress.
```

A user who issues `CREATE DBSPACE` during a drop sees:

```
Cannot perform requested command as there is a
DROP DBSPACE command in progress.
```

A user who issues `DROP DBSPACE` during a create sees:

```
Cannot perform requested command as there is a
CREATE DBSPACE command in progress.
```

When one transaction issues a DDL command on a given table or index, any other transaction that began before the DDL transaction commits, and that tries to access that table, receives an error. When this error occurs, any additional attempts to read or write to the table in the current transaction fail.

There is an exception to these index creation commands rules. If you use a `CREATE INDEX` command with a `SELECT` statement on the table(s) affected by the index creation, SAP IQ prevents use of the new index until the transaction creating the index commits.

While `GRANT`, `REVOKE`, and `SET OPTION` are also considered DDL operations, they cause no concurrency conflicts, and so are not restricted. `GRANT` and `REVOKE` always cause an automatic commit; `SET OPTION` causes an automatic commit except when it is specified as `TEMPORARY`. `GRANT` and `REVOKE` are not allowed for any user currently connected to the database. `SET OPTION` affects all subsequent SQL statements sent to the database server, except for certain options that do not take effect until after you restart the database server.

10.3.1.2.3 Primary Keys and Locking

Because only one user can update a table when table-level snapshot versioning is enabled, primary key generation does not cause concurrency conflicts.

10.3.1.3 Displaying Active Table Locks

Identify the user who has a table locked. An attempt to write to a table fails if another transaction holds a lock on that table.

Procedure

1. Run `sp_iqtransaction`, and find the transaction identifier and user name in the output.
2. Run the `sp_iqlocks` procedure.

   For each lock in the catalog store and the IQ store of your current database, `sp_iqlocks` returns the following information:
   - The connection and user ID that holds the lock
   - The table on which the lock is held
   - The type of lock, and a name to identify the lock
3. When another transaction holds a lock, you see a message similar to the following:

   ```
   Cannot open the requested object for write in the current
   transaction (TxnID1). Another user has write access in transaction
   TxnID2.
   ```

   Find `TxnID2` in the output of `sp_iqtransaction`, and look for the name of the user in the same row of output.

10.3.1.4 Performance Implications of Table-Level Snapshot Versioning

Typically, table-level snapshot versioning has a minimal impact on performance.

However, there are certain resource issues you should be aware of:

- Buffer consumption may increase slightly if multiple users are using different versions of the same database page simultaneously.
- Version management requires some overhead, but the effect on performance is minimal.
- The thread control, which determines how many processing resources a user gets, and the sweeper controls, which use a small number of threads to sweep dirty data pages out to disk, have a minor impact on performance.
- Disk space can sometimes become an issue. Storing overlapping versions has the potential to use a lot of disk space, depending on the number and size of versions in use simultaneously. Metadata and database

page versions are retained until they are dropped, either at a `RELEASE SAVEPOINT` or when the last transaction that can see a given version commits or rolls back. The space is then reclaimed.

Delays due to locking are minimal. Individual commits, rollbacks, and checkpoints might block other read or write transactions, but only very briefly.

All of these performance and disk use factors only affect your system in the degree to which you take advantage of the SAP IQ concurrent read and write capabilities. Disk space requirements in particular can vary widely, depending on how long write transactions take before they commit, how many read transactions take place during write transactions, the number of rows these transactions affect, and whether you allow the release of data pages at interim savepoints.

In this section:

Overlapping Versions and Deletions [page 223]
To delete data when using table-level versioning, you may actually need to increase disk space by adding a dbspace to your IQ main store.

10.3.1.4.1 Overlapping Versions and Deletions

To delete data when using table-level versioning, you may actually need to increase disk space by adding a dbspace to your IQ main store.

The amount of space you need for a deletion depends on the distribution of the data on data pages, more than on the size or number of rows being deleted. SAP IQ must retain a version of each page that contains any of the data you are deleting, from the time the deletion begins until the transaction commits. If the rows being deleted are distributed across many data pages, you need space in your IQ main store to retain all of those extra data pages.

For example, say you must delete 10 rows from a database where each page holds 100 rows. If each of those 10 rows is on a separate data page, then your IQ main store must have space for 10 version pages, each big enough to hold 100 rows. This distribution is unlikely, but it is possible.

The space needed to delete data varies by index type. It is proportional to—and in the worst case, equal to—the size of the index from which you are deleting.

If you run out of space while deleting data, SAP IQ halts the deletion and writes this message in the notification log:

```
Out of disk space
```

After you add space, the deletion resumes. When the delete transaction commits, the space becomes available for other deletions or insertions. If you do not normally need that much space in your database, you can drop the dbspace to regain the extra disk space for other purposes. Do so before inserting any data that may need the new dbspace.

Running out of space during a deletion should not affect other query users.

If you run out of space, but do not have enough disk space to add another dbspace, you must shut down the database engine and then restart it, allowing the database to roll back. You can then delete the rows in smaller, separate transactions.

> **i Note**
>
> `DROP TABLE` and `DROP DATABASE` delete the table or database and all data in it without creating any version pages, so you do not need to add space to use these commands.

10.3.2 Row-Level Snapshot Versioning

Row-level snapshot versioning applies only to tables enabled for in-memory RLV storage. Row-level snapshot versioning allows multiple writers to make concurrent DML changes to a table, but never to the same rows at the same time.

Row-level snapshot versioning locks the table at the row level using row locks. A row lock provides a write lock for a table row, meaning the transaction gets blocked, or fails, depending on the `BLOCKING` and `BLOCKING_TIMEOUT` option settings. If `BLOCKING` is ON, the transaction blocks. If `BLOCKING` is OFF, the transaction fails immediately with an `ALREADY LOCKED` SQL exception.

Transaction blocking enables row-level snapshot versioning to write to different rows of the same table simultaneously. Depending on the `BLOCKING` and `BLOCKING_TIMEOUT` option settings, row-lock contention results either in an error, or a retry to obtain the lock if it is released within the specified timeout period. When a transaction configured for table-level versioning attempts to write to a table with a row locked by a row-level versioned transaction, the table-level transaction either fails with an error, or blocks and retries if the lock is released within the specified timeout period.

DDL changes to a table (`CREATE`, `DROP`, and `ALTER`), however, lock the table at the table level.

In this section:

Specifying Snapshot Versioning [page 225]
: Use the `SNAPSHOT_VERSIONING` option to set the snapshot versioning type to either `Row-level` or `Table-Level`. You can set the option at the database (PUBLIC) level, connection level (TEMPORARY) or user level. To use the in-memory RLV store, enable row-level snapshot versioning for your transactions. For simultaneous updates to different rows of the same table, each transaction or connection must also enable row-level snapshot versioning

Tutorial: Monitoring Write-Intent Locks [page 226]
: In this tutorial, create RLV-enabled tables, execute a transaction, and use the `sp_iqlocks` stored procedure to report on schema-level locks and write-intent locks in the database. Then use the `sp_iqconnection` and `sa_conn_info` stored procedures to view the internal connection controlling the write-intent lock.

Monitoring Write-Intent Locks in Interactive SQL [page 231]
: View details on write intent locks using the `sa_locks` system procedure.

Row-Level DDL Locking Considerations [page 232]
: Data Definition Language (DDL) changes (for example, `CREATE INDEX`, `DROP INDEX`, and `ALTER TABLE` ADD, ALTER, or DROP) to an RLV-enabled table require an exclusive table-level lock. For DDL events, the locking behavior for an RLV-enabled table is the same as for an IQ main store table: the writing connection has an exclusive lock on the table. When `BLOCKING` is set to ON, all competing DML and DDL transactions against the table are blocked until the DDL changes are committed. When `BLOCKING` is set to OFF, the competing transaction will immediately fail the lock request.

10.3.2.1 Specifying Snapshot Versioning

Use the `SNAPSHOT_VERSIONING` option to set the snapshot versioning type to either `Row-level` or `Table-Level`. You can set the option at the database (PUBLIC) level, connection level (TEMPORARY) or user level. To use the in-memory RLV store, enable row-level snapshot versioning for your transactions. For simultaneous updates to different rows of the same table, each transaction or connection must also enable row-level snapshot versioning

Prerequisites

- If setting to Row-level, the RLV store dbspace exists with at least one dbfile.
- If setting to Row-level, the table is RLV-enabled.
- Requires the SET ANY PUBLIC OPTION system privilege to set this option for PUBLIC or for other user or role.

Context

Once the snapshot versioning property has been set for a transaction, it remains the same until the transaction commits.

Procedure

1. Determine the scope of the `SET OPTION` command to set the option as a database-wide option, connection-level option, or user-level option:
   - `SET OPTION PUBLIC.SNAPSHOT_VERSIONING...`
   - `SET TEMPORARY OPTION SNAPSHOT_VERSIONING...`
   - `SET OPTION <username>.SNAPSHOT_VERSIONING...`
2. Specify the snapshot versioning type.

Level	Option
Row-level	Row-level snapshot versioning. Required for in-memory RLV storage. Row-level snapshot versioning allows multiple writers to make concurrent DML changes to a table, but never to the same rows at the same time.
Table-level	Classic (backward-compatible) SAP IQ versioning behavior. Takes snapshots at the table-level. Multiple writers cannot make concurrent DML changes to a table.

Example

```
SET TEMPORARY OPTION SNAPSHOT_VERSIONING = 'Row-level';
CREATE TABLE rv_locks(c1 int, c2 int, c3 int) ENABLE RLV STORE;
INSERT INTO rv_locks VALUES (1,1,1);
INSERT INTO rv_locks VALUES (2,2,2);
INSERT INTO rv_locks VALUES (3,3,3);
INSERT INTO rv_locks VALUES (4,4,4);
COMMIT;
```

10.3.2.2 Tutorial: Monitoring Write-Intent Locks

In this tutorial, create RLV-enabled tables, execute a transaction, and use the `sp_iqlocks` stored procedure to report on schema-level locks and write-intent locks in the database. Then use the `sp_iqconnection` and `sa_conn_info` stored procedures to view the internal connection controlling the write-intent lock.

Prerequisites

- You are connected as the DBA user. This ensures you have the required privileges to complete each task in the tutorial.
- The connection is new; no work has been performed in it.
- RLV storage is configured.

Context

> → Tip
>
> You can monitor locks and deadlocks using SAP IQ Cockpit. For more information, see *Monitor SAP IQ* in the *SAP IQ Cockpit* documentation.

Procedure

1. Enable table-level snapshot versioning for the current session:

   ```
   SET TEMPORARY OPTION SNAPSHOT_VERSIONING = 'table-level';
   ```

2. (Skip this step if RLV-enabled tables `rv_locks` and `rv_locks2` already exist) Create RLV-enabled tables `rv_locks` and `rv_locks2`:

   ```
   CREATE TABLE rv_locks(c1 INT, c2 INT, c3 INT);
   INSERT INTO rv_locks VALUES (1,1,1);
   ```

```
INSERT INTO rv_locks VALUES (2,2,2);
INSERT INTO rv_locks VALUES (3,3,3);
INSERT INTO rv_locks VALUES (4,4,4);
COMMIT;
CREATE TABLE rv_locks2(c1 int, c2 int, c3 int);
INSERT INTO rv_locks2 VALUES (1,1,1);
INSERT INTO rv_locks2 VALUES (2,2,2);
INSERT INTO rv_locks2 VALUES (3,3,3);
INSERT INTO rv_locks2 VALUES (4,4,4);
COMMIT;
ALTER TABLE rv_locks ENABLE RLV STORE;
ALTER TABLE rv_locks2 ENABLE RLV STORE;
```

3. Enable connection blocking and set the blocking timeout threshold:

```
SET TEMPORARY OPTION BLOCKING = 'ON';
SET TEMPORARY OPTION BLOCKING_TIMEOUT = '0';
```

4. Execute `sp_iqlocks` to view the current set of database locks:

```
sp_iqlocks
```

   At this point, no locks are returned. The absence of a write-intent lock for the RLV-enabled table indicates that the in-memory RLV portion of the table has yet to be created.

5. Perform a commit:

```
COMMIT
```

6. Set the snapshot versioning property of the transaction to row-level:

```
SET TEMPORARY OPTION SNAPSHOT_VERSIONING = 'row-level';
```

7. Execute a write to the table:

```
INSERT INTO rv_locks VALUES (5,5,5);
```

   Writing to, or querying, an RLV-enabled table creates the RLV-enabled portion of the table in memory, on demand.

8. Re-execute `sp_iqlocks`:

```
sp_iqlocks
```

   This time, the procedure returns a write-intent lock:

conn_name	conn_id	user_id	table_type	creator	table_name
SQL_DBC_13cd6038	3	DBA	BASE	DBA	rv_locks
RLV_CONN_T775	1000000407	DBA	BASE	DBA	rv_locks

index_id	lock_class	lock_duration	lock_type	row_identifier	row_range
(NULL)	Schema	Transaction	Shared	(NULL)	(NULL)
(NULL)	Table	Transaction	Intent	(NULL)	(NULL)

The connection ID number (conn_id) 100000407 has a write-intent lock on the `rv_locks` table. The lock type is set to `Intent`, which indicates a write intent lock.

> **i Note**
>
> The connection ID number (100000407) is large because it represents an internal connection within the server itself. This internal connection is used to manage locks on the RLV-enabled table.

The connection ID number (conn_id) 3 has a schema lock on the table. The lock type is set to `Shared`, which indicates a shared-schema lock.

9. Commit the transaction that performed the insert:

```
COMMIT
```

During the commit, the database releases the locks held by the transaction. For the tutorial, this releases only the shared-schema lock. The RLV-enabled table now exists in memory, with committed data. Therefore, the only lock present at this point is the write-intent lock held by the RLV-enabled portion of the table.

10. Re-execute `sp_iqlocks`:

```
sp_iqlocks
```

The schema lock is gone, but the write-intent lock remains:

conn_name	conn_id	user_id	table_type	creator	table_name
RLV_CONN_T775	1000000407		BASE	DBA	rv_locks

index_id	lock_class	lock_duration	lock_type	row_identifier	row_range
(NULL)	Table	Transaction	Intent	(NULL)	(NULL)

> **i Note**
>
> The row for connection ID number (conn_id) 100000407 has not changed since the last time you executed `sp_iqlocks`.

11. Execute `sp_iqconnection` to view connection details:

```
sp_iqconnection
```

You see the following:

ConnHandle	Name	Userid	LastReqTime	ReqType	IQCmdType
1	SQL_DBC_13de5fd8	DBA	2012-08-08 08:49:25.629	PREFETCH	NONE
3	SQL_DBC_13cd6038	DBA	2012-08-08 09:25:32.920	OPEN	IQUTILITYOPEN-CURSOR
1000000407	INT: RLVLock-Conn			unknown (0)	NONE

LastIQCmdTime	IQCursors	LowestIQCursor-State	IQthreads	TxnID	ConnCreateTime
2012-08-08 08:49:25.0	0	NONE	0	0	2012-08-08 08:49:24.0
2012-08-08 09:25:32.0	0	NONE	0	1008	2012-08-08 08:50:04.0
0001-01-01 00:00:00.0	0	NONE	0	0	2012-08-08 09:00:40.0

TempTableSpaceKB	TempWorkSpaceKB	IQconnID	satoiq_count	iqtosa_count	CommLink
0	0	70	40	2	local
0	0	92	187	413	local
0	0	410	2	0	NA

NodeAddr	LastIdle	MPXServerName	LSName	INCConnName	INCConnSuspended
	0				N
	8789				N
NA	0				N

The third row (ConnHandle 1000000407) provides information on the internal connection (RLVLockConn) used by the RLV-enabled table to control the write-intent lock.

> **i Note**
>
> ConnHandle 1000000407 matches conn_id 100000407 in `sp_iqlocks` output. It also matches ConnHandle 1000000407 in `sp_iqtransaction` output.

12. Execute `sa_conn_info` to view additional connection details:

```
sa_conn_info
```

You see the following:

Number	Name	Userid	DBNumber	LastReqTime	ReqType
1000000407	sa_INT: RLVLock-Conn		0		unknown (0)
3	SQL_DBC_13cd6038	DBA	0	2012-08-08 09:30:43.799	FETCH
1	SQL_DBC_13de5fd8	DBA	0	2012-08-08 08:49:25.629	PREFETCH

CommLink	NodeAddr	ClientPort	ServerPort	BlockedOn
NA	NA	0	0	0

CommLink	NodeAddr	ClientPort	ServerPort	BlockedOn
local		0	0	0
local		0	0	0

LockRowID	LockIndexID	LockTable	UncommitOps	ParentConnection
0			0	
0			0	
0			0	

In the first row, Number 1000000407 matches ConnHandle 1000000407 in the `sp_iqconnection` output, and conn_id 100000407 in the `sp_iqlocks` output.

Userid "INT: RLVLockConn" indicates an internal connection. This connection is used by the RLV-enabled table to control the write-intent lock.

`sa_conn_info` is similar to `sp_iqconnection`.

In this section:

Row Locks [page 230]
A row lock is a table-row write lock that allows the holding transaction to write to any column of a locked row. Only one holder of this lock can exist at a time. A write-intent lock is a prerequisite; the transaction must hold a write-intent lock before the lock manager grants it a row lock.

10.3.2.2.1 Row Locks

A row lock is a table-row write lock that allows the holding transaction to write to any column of a locked row. Only one holder of this lock can exist at a time. A write-intent lock is a prerequisite; the transaction must hold a write-intent lock before the lock manager grants it a row lock.

A table-row write lock allows the holding transaction to write to any column of locked row. This lock cannot be granted without the requesting transaction first holding the write intent lock. Row write locks are exclusive locks; only one transaction can hold a write lock on a row at any time. Once a transaction acquires a write lock, requests to lock the row by other transactions are denied.

Row locks exist only during row deletions. The RLV store is an append-only store, meaning that every write action results in a new row appended to the store. `INSERT` statements append a new row to the store, as do `UPDATE` statements. The RLV store considers an `UPDATE` to be a `DELETE` followed by an `INSERT`. Before a row is deleted, either in the context of a `DELETE` or `UPDATE` statement, the database takes out a row-level lock.

10.3.2.3 Monitoring Write-Intent Locks in Interactive SQL

View details on write intent locks using the `sa_locks` system procedure.

Prerequisites

- RLV storage is configured.
- A table is registered for RLV storage.
- The `SNAPSHOT_VERSIONING` database option is set to 'Row-level.'

Procedure

1. Run the `sa_locks` system procedure.
2. View the `lock_class` column, which shows intent locks for tables and rows.
3. Run the `sp_iqlocks` stored procedure to identify users who are blocking other users from writing to a table.

   This procedure displays information about locks currently held in the database, including the connection and user ID that holds the lock, the table on which the lock is held, the type of lock, and a name to identify the lock.

In this section:

Write-Intent Locks [page 231]
A write-intent lock is a table write lock that grants the transaction permission to write to a table row in the future. A write-intent lock can be held by multiple requesting connections.

10.3.2.3.1 Write-Intent Locks

A write-intent lock is a table write lock that grants the transaction permission to write to a table row in the future. A write-intent lock can be held by multiple requesting connections.

A write intent lock always exists when the RLV-enabled portion of the table exists in memory. You can view details of the write intent lock using the `sp_iqlocks` stored procedure.

The write-intent lock conflicts with table write locks and table exclusive locks. This conflict prevents a table-level snapshot-versioned transaction from writing to the table or performing a DDL operation until the lock manager releases all write-intent locks on the table. In a situation where both table-level snapshot-versioned

transactions and row-level snapshot-versioned transaction connections write to a table, write-intent locks provide synchronization. Consider this scenario:

Connection	Action
Row-level snapshot-versioned transaction A	• Executes query writing to multiple rows of table_1. • Lock manager creates a write-intent lock for table_1. • Lock manager creates multiple local write-intent locks for row-level DML updates. Lock manager creates row-level locks.
Table-level snapshot-versioned transaction B	Attempts to write to table_1. Transaction B blocked by write intent lock.
Row-level snapshot-versioned transaction A	Commits transaction A. Table changes are merged from the RLV store to the IQ main store. Write-intent locks released.
Table-level snapshot-versioned transaction B	Proceeds with write to table_1.

10.3.2.4 Row-Level DDL Locking Considerations

Data Definition Language (DDL) changes (for example, `CREATE INDEX`, `DROP INDEX`, and `ALTER TABLE` ADD, ALTER, or DROP) to an RLV-enabled table require an exclusive table-level lock. For DDL events, the locking behavior for an RLV-enabled table is the same as for an IQ main store table: the writing connection has an exclusive lock on the table. When `BLOCKING` is set to ON, all competing DML and DDL transactions against the table are blocked until the DDL changes are committed. When `BLOCKING` is set to OFF, the competing transaction will immediately fail the lock request.

10.3.3 Viewing the Versioning Type of a Transaction

Use the `sp_iqtransaction` stored procedure to view the versioning type (either table-level or row-level) of each active transaction.

Procedure

1. Run `sp_iqtransaction`.

   See *sp_iqtransaction Procedure* in SAP IQ SQL Reference for complete syntax information.
2. Examine the VersioningType column.

VersioningType	Description
Row-level	The transaction uses row-level versioning. Row-level versioning enables concurrent writer access and row-level locking for RLV-enabled tables.

VersioningType	Description
Table-level	The transaction uses table-level versioning. Table-level versioning enables single-writer access and table-level locking.

10.3.4 Temporary Table Versioning

A temporary table that is created in the database is called a global temporary table. A temporary table that is declared (rather than created in the database) is called a local temporary table.

A global temporary table is accessible to all users with the appropriate permissions. Each user has his or her own instance of the table, however; only one user ever sees a given set of rows. By default, the rows of a global temporary table are deleted on `COMMIT`. You can override this default, by specifying `ON COMMIT PRESERVE ROWS` when you create the temporary table.

Only one user sees any of the rows in a local temporary table. The table is dropped when that user disconnects. When you declare a local temporary table, SAP IQ issues a savepoint instead of committing the transaction automatically, as it would for a data definition operation on any other type of table. Before creating an index, commit the data in the local temporary table. If you attempt to create an index using uncommitted data, you may see:

```
Local temporary table, <tablename>,
must be committed in order to create an index.
```

SAP IQ makes no distinction between versioning base tables (IQ main store database tables) and versioning global temporary tables. Because the data in any temporary table is accessible to only one user, there will never be more than one write transaction open for a temporary table.

You can enable a global temporary table or local temporary table for RLV storage, even though a temporary table cannot take advantage of RLV's multiple user concurrent write capability. Enabling a temporary table for RLV storage lets you take advantage of the RLV store's low-latency DML.

10.4 Investigating Lock Contention Effects on Performance

Some load or query performance issues result from lock contention. If your kernel system time is greater than 10%, you may be experiencing lock contention.

Procedure

- To find out if lock contention is affecting performance on your system, do one of the following:
  - Run the SAP IQ monitor with the `-contention` option.
  - (UNIX platforms) Run the `sar` or `vmstat` utility.
  - (Windows platforms) Check the CPU usage in the Task Manager.
- If you suspect lock contention, you may find it useful to control the level of partitioning directly by setting:

- The `-iqpartition` server startup option.
- The `CACHE_PARTITIONS` database option.

> **i** Note
>
> Higher-than-normal kernel system time can also indicate that your kernel is not well-tuned. If this is the case, you may need to adjust kernel parameters; changing SAP IQ settings does not fix an improperly tuned kernel.

In this section:

(Deprecated) -contention [page 234]
Displays many key buffer cache and memory manager locks. These lock and mutex counters show the activity within the buffer cache and heap memory and how quickly these locks were resolved. Timeout numbers that exceed 20% indicate a problem.

10.4.1 (Deprecated) -contention

Displays many key buffer cache and memory manager locks. These lock and mutex counters show the activity within the buffer cache and heap memory and how quickly these locks were resolved. Timeout numbers that exceed 20% indicate a problem.

Usage

```
monitor_options -contention
```

Output

Output Field	Description
`<AU>`	Current number of active users
`<LRULks>`	Number times the LRU was locked (repeated for the temp cache)
`<woTO>`	Number times lock was granted without timeout (repeated for the temp cache)
`<Loops>`	Number times SAP IQ retried before lock was granted (repeated for the temp cache)
`<TOs>`	Number of times SAP IQ timed out and had to wait for the lock (repeated for the temp cache)

Output Field	Description
<BWaits>	Number of busy waits for a buffer in the cache (repeated for the temp cache)
<IOLock>	Number of times SAP IQ locked the compressed I/O pool (repeated for the temp cache); can be ignored
<IOWait>	Number of times SAP IQ had to wait for the lock on the compressed I/O pool (repeated for the temp cache); can be ignored
<HTLock>	Number of times SAP IQ locked the block maps hash table (repeated for the temp cache)
<HTWait>	Number of times SAP IQ had to wait for the block maps hash table (repeated for the temp cache); HTLock and HTWait indicate how many block maps you are using
<FLLock>	Number of times SAP IQ had to lock the free list (repeated for the temp cache)
<FLWait>	Number of times SAP IQ had to wait for the lock on the free list (repeated for the temp cache)
<MemLks>	Number of times SAP IQ took the memory manager (heap) lock
<MemWts>	Number of times SAP IQ had to wait for the memory manager lock

> **i** Note
>
> SAP IQ no longer uses spin locks. As a result, woTO, Loops, and TOs statistics are rarely used.

10.5 Checkpoints, Savepoints, and Transaction Rollback

Besides permitting concurrency, snapshot versioning transaction processing plays an important role in data recovery.

SAP IQ relies on transaction-related commands that help you recover a stable set of data in the event of system or media failure. The `CHECKPOINT` command sets a checkpoint, the `SAVEPOINT` command sets a savepoint, `RELEASE SAVEPOINT` releases savepoints, and `ROLLBACK TRANSACTION` rolls back transactions.

In this section:

Checkpoints [page 236]
A checkpoint marks a significant point in a transaction, when SAP IQ writes to disk certain information it tracks internally. SAP IQ uses this information during database recovery.

Savepoints Within Transactions [page 236]
SAP IQ supports savepoints within a transaction.

Transaction Rollback [page 238]

When you roll back a transaction, either automatically, or by explicit user request, you undo all of the operations in that transaction.

System Recovery [page 239]
In the event of a system failure or power outage, or when you restart the database server after it has been stopped, SAP IQ attempts to recover automatically.

How Transaction Information Aids Recovery [page 240]
The SAP IQ recovery mechanism is designed for a data warehouse environment in which transactions are typically few but lengthy.

Concurrency for Backups [page 240]
Backups may be performed concurrently with read and write operations.

10.5.1 Checkpoints

A checkpoint marks a significant point in a transaction, when SAP IQ writes to disk certain information it tracks internally. SAP IQ uses this information during database recovery.

SAP IQ uses checkpoints differently than OLTP databases such as SAP SQL Anywhere. OLTP databases tend to have short transactions that affect only a small number of rows. Writing entire pages to disk would be very expensive for them. Instead, OLTP databases generally write to disk at checkpoints, and write only the changed data rows.

SAP IQ is an OLAP database. A single OLAP transaction can change thousands or millions of rows of data. For this reason, the database server does not wait for a checkpoint to occur to perform physical writes. It writes updated data pages to disk after each transaction commits. For an OLAP database, writing full pages of data to disk is much more effective than writing small amounts of data at arbitrary checkpoints.

Most checkpoints occur automatically. You can also set explicit checkpoints, although you do not need to do so.

A checkpoint occurs:

- When a transaction issues a `CHECKPOINT` command
- When the `CHECKPOINT_TIME` is exceeded
- At the start and end of the backup process
- When the database server is shut down

The `CHECKPOINT_TIME` is the maximum time that can pass between checkpoints. By default, it is 60 minutes. To adjust the checkpoint interval, use the `SET OPTION` statement. Adjusting the checkpoint time or issuing explicit checkpoints may be unnecessary. Controlling checkpoints is less important in SAP IQ than in OLTP database products, because SAP IQ writes the actual data pages after each transaction commits.

10.5.2 Savepoints Within Transactions

SAP IQ supports savepoints within a transaction.

A `SAVEPOINT` statement defines an intermediate point during a transaction. Because a single IQ transaction may write millions of rows of data, you may want to limit the amount of data that is committed—and thus

written to the IQ main store—to less than a full transaction's worth. Setting savepoints allows you to subdivide transactions.

In this section:

Releasing a Savepoint [page 237]
Releasing a savepoint frees up the version pages that have been used up to that savepoint, and makes better use of your disk space.

Rollbacks to Savepoints [page 238]
A `ROLLBACK TO SAVEPOINT` command undoes all changes after a savepoint.

Automatic and User-Defined Savepoints [page 238]
IQ sets an implicit savepoint before and after every DML command.

Named and Nested Savepoints [page 238]
Named, nested savepoints provide many active savepoints within a transaction.

10.5.2.1 Releasing a Savepoint

Releasing a savepoint frees up the version pages that have been used up to that savepoint, and makes better use of your disk space.

Context

Data is versioned internally at the page level. SAP IQ maintains a separate copy of only the updated pages; the remaining pages are shared with the previous version.

Procedure

Run the RELEASE SAVEPOINT statement, where `<savepoint-name>` is a specific identifier:

```
RELEASE SAVEPOINT <savepoint-name>
```

The most recent savepoint is released if you omit `<savepoint-name>`.

All resources after the savepoint are released, and you cannot roll back to any intermediate savepoints.

> **i** Note
>
> `RELEASE SAVEPOINT` does not release locks.

10.5.2.2 Rollbacks to Savepoints

A `ROLLBACK TO SAVEPOINT` command undoes all changes after a savepoint.

This command rolls back to the savepoint you specify, or to the most recent savepoint if you do not specify a named savepoint. Rolling back to savepoint `<n>` undoes all actions for all savepoints greater than or equal to `<n>`.

Normally, locks are released only at the end of a transaction. However, `ROLLBACK TO SAVEPOINT` does release locks under certain conditions, as in the following scenario:

Assume you have a series of savepoints in a transaction, and you then perform a write operation. Roll back the transaction to an earlier savepoint. The rollback undoes all actions after that savepoint, including the write operation and any locks it acquires after the savepoint you are rolling back to.

SAP IQ supports savepoint operations on updatable cursors.

10.5.2.3 Automatic and User-Defined Savepoints

IQ sets an implicit savepoint before and after every DML command.

The data page versions associated with these savepoints are released when the command completes. To retain data page versions beyond the end of a single DML command, set your own, named savepoints.

10.5.2.4 Named and Nested Savepoints

Named, nested savepoints provide many active savepoints within a transaction.

You can cancel changes between a `SAVEPOINT` and a `RELEASE SAVEPOINT` by rolling back to a previous savepoint or rolling back the transaction itself. Changes within a transaction are not a permanent part of the database until the transaction is committed. All savepoints are released when a transaction ends.

Savepoints cause SAP IQ to update information it maintains about the location of available disk space. This information is used during transaction rollback.

There is no additional overhead in using savepoints, although unreleased savepoints may consume extra disk space by keeping older intermediate versions active.

10.5.3 Transaction Rollback

When you roll back a transaction, either automatically, or by explicit user request, you undo all of the operations in that transaction.

Rolling back the database means returning the database to an earlier state.

In this section:

Cause of Rollback [page 239]
Rollbacks can occur either due to an explicit user request, or automatically.

Effect of Rollback [page 239]
Rollback returns both the main and temporary stores to their former state, and also releases locks.

10.5.3.1 Cause of Rollback

Rollbacks can occur either due to an explicit user request, or automatically.

Use a `ROLLBACK` statement to undo any changes to the database since the last `COMMIT` or `ROLLBACK`.

Use a `ROLLBACK TO SAVEPOINT` statement to undo any changes to the database since the `SAVEPOINT` you name, or else to the last `SAVEPOINT`.

SAP IQ rolls back the database automatically if a user is in a transaction and then logs out or disconnects without committing. The rollback is to the most recent commit or rollback.

10.5.3.2 Effect of Rollback

Rollback returns both the main and temporary stores to their former state, and also releases locks.

- Transaction rollback releases all locks held by the transaction.
- Rollback to a savepoint releases all locks acquired after that savepoint.

Rolling back open cursors deletes all cursor information and closes both hold and non-hold cursors:

- Transaction rollback closes all cursors, regardless of whether the cursor was opened in the transaction being rolled back, or in an earlier transaction.
- Rollback to a savepoint closes all cursors opened after that savepoint.

10.5.4 System Recovery

In the event of a system failure or power outage, or when you restart the database server after it has been stopped, SAP IQ attempts to recover automatically.

During SAP IQ database recovery, any uncommitted transactions are rolled back, and any disk space used for old versions is returned to the pool of available space. At this point, the database contains only the most recently committed version of each permanent table.

During recovery from a system failure, SAP IQ reopens all connections that were active at the time of the failure. If the -gm parameter, which sets the number of user connections, was in effect at the time of the failure, restart the IQ server with at least as many connections as were in use when the failure occurred. Temporary table contents cannot be recovered.

If a failure occurs, try to restart the database server and database. You will need information from your server log and IQ message log to recover.

Run the `sp_iqcheckdb` stored procedure after a system failure, preferably before allowing users to connect. This procedure checks every block in your database, and produces statistics that allow you to check the consistency and integrity of your database.

10.5.5 How Transaction Information Aids Recovery

The SAP IQ recovery mechanism is designed for a data warehouse environment in which transactions are typically few but lengthy.

SAP IQ performs database updates on a copy of the actual database page, and then writes the data to disk whenever a write transaction commits. SAP IQ also records:

- The location and quantity of changed data for each transaction is stored in a transaction log.
- The location of any version pages and free space is stored on disk. SAP IQ uses this information to free up space when versions are no longer needed. All versions created during a write transaction become obsolete when the write transaction commits or rolls back. Individual versions can be released at a savepoint.
- Additional information about checkpoints that occurred during a transaction.

When you need to recover your database, SAP IQ restores quickly from the information in the transaction log and the checkpoint information. It uses the information about versions and free space to roll back transactions, and to release the disk space occupied by obsolete versions.

The transaction log requires very little space — only about 128 bytes for each committed transaction. The space requirements for checkpoint and disk space availability information are also very small. In systems with a high number of transactions that change data, however, the transaction log can grow to be very large, requiring periodic truncation.

The checkpoint information is deleted at the next checkpoint. Information related to particular savepoints is deleted when the savepoint is released or rolled back.

10.5.6 Concurrency for Backups

Backups may be performed concurrently with read and write operations.

Backup is a DML operation. Backup backs up as of the start of the backup command (the checkpoint). Restore operations, however, require exclusive access, because they write to the database.

10.6 Cursors in Transactions

Return the results of a `SELECT` in the form of a data type called a cursor.

A cursor is similar to a table, but one row is identified as the present, or current row. Various commands allow you to navigate through the rows of a cursor. For example, the `FETCH` command retrieves a row from the cursor and identifies it as the current row. You can step through all the rows in a cursor by calling this command repeatedly.

You may find cursors useful when you write procedures, or applications that access a database using Embedded SQL. They are also used by many front-end query tools. They are unavailable when you use Interactive SQL.

SAP IQ cursors are updatable, which allows you to modify the underlying data in the database while processing a cursor.

The rows in a cursor, like those in a table, have no order associated with them. The `FETCH` command steps through the rows, but the order may appear random and inconsistent. Impose an order by appending an `ORDER BY` phrase to your `SELECT` statement.

The `sp_iqcursorinfo` stored procedure displays information about open cursors on the server.

In this section:

Cursors and Versioning [page 241]
When you use cursors, SAP IQ must be able to manage multiple versions within a single transaction.

Cursor Sensitivity [page 242]
A cursor is said to be sensitive if its membership—the data rows it returns—can vary from the time it is opened until the time it is closed. An insensitive cursor has its membership fixed when it is opened.

Cursor Scrolling [page 242]
SAP IQ cursors can be either scrolling or non-scrolling.

Hold Cursors [page 242]
Specifying the `HOLD` option when you open a cursor keeps the cursor open past the end of the transaction, if the transaction ends in a `COMMIT`.

Positioned Operations [page 243]
In a positioned operation, the current location of the cursor determines where a read or write operation begins.

Message Logging for Cursors [page 243]
By default, cursor operations are not logged in the SAP IQ message file.

10.6.1 Cursors and Versioning

When you use cursors, SAP IQ must be able to manage multiple versions within a single transaction.

For example, assume that you open a cursor called `cust_cursor` at time `<x>` that uses the `customer` table. You update that table at time `<y>`. SAP IQ retains the version of the `customer` table from time `<x>` until you are done using `cust_cursor`.

The support of cursors by SAP IQ is oriented toward their likely use in decision support system applications.

10.6.2 Cursor Sensitivity

A cursor is said to be sensitive if its membership—the data rows it returns—can vary from the time it is opened until the time it is closed. An insensitive cursor has its membership fixed when it is opened.

The membership and values of the result set of an insensitive cursor are indeterminate with respect to changes. A value-sensitive cursor is insensitive with respect to its membership and sensitive with respect to the order and values of the result set. SAP IQ supports insensitive updatable cursors.

10.6.3 Cursor Scrolling

SAP IQ cursors can be either scrolling or non-scrolling.

Non-scrolling cursors allow only the command forms `FETCH NEXT` and `FETCH RELATIVE 0` to find and retrieve data. They do not keep track of which rows have been fetched. A cursor declared as `DYNAMIC SCROLL` is the same as a cursor declared as `SCROLL`.

Set the `FORCE_NO_SCROLL_CURSORS` option on to save on temporary storage requirements if you are retrieving very large numbers (millions) of rows. However, if your front-end application makes frequent use of backward-scrolling cursor operations, query response is faster with this option off.

If your front-end application rarely performs backward scrolling, make `FORCE_NO_SCROLL_CURSORS = 'ON'` a permanent `PUBLIC` option. It uses less memory and improves query performance.

10.6.4 Hold Cursors

Specifying the `HOLD` option when you open a cursor keeps the cursor open past the end of the transaction, if the transaction ends in a `COMMIT`.

A hold cursor does not remain open across a `ROLLBACK` in which a cursor is opened.

In SAP IQ, hold cursors can be updated until they are committed. After it is committed, a hold cursor is marked internally as read-only and subsequent positioned updates generate an error.

Although the `HOLD` option is not commonly used in a DSS environment, it may prove useful in some situations. For example, many existing applications expect to use hold cursors, and some ODBC drivers use hold cursors by default.

SAP IQ provides the version management needed for hold cursors.

Hold cursors do impact performance. All resources used by the cursor, including memory, disk space, and process threads, are held until the cursor is closed.

10.6.5 Positioned Operations

In a positioned operation, the current location of the cursor determines where a read or write operation begins.

SAP IQ supports positioned fetches, which can be helpful in long query transactions. SAP IQ also supports positioned update and delete operations, which are intended for shorter insertions and deletions. For the most part, updates to SAP IQ databases are likely to involve large amounts of data; repositioning is a very minor part of such write operations.

Positioned updates and deletes are handled as operations on the cursor, and therefore part of its transaction, rather than as separate statements. Any failure that occurs after the cursor is open results in a rollback of all operations that have been performed through this open cursor.

10.6.6 Message Logging for Cursors

By default, cursor operations are not logged in the SAP IQ message file.

To track cursor operations to determine the cause of a problem, turn on the `LOG_CURSOR_OPERATIONS` option to produce a message each time a cursor is opened or closed. Data changes made through an updatable cursor are also logged in the IQ message file.

10.7 Remote Transactions

Transaction management involving remote servers uses a two-phase commit protocol.

SAP IQ implements a strategy that ensures transaction integrity for most scenarios.

In this section:

Remote Transaction Restrictions [page 243]
Remote transaction management has savepoints and nested statement restrictions.

Remote transaction management and restrictions [page 244]
A multi-phase commit protocol is used for managing transactions that involve remote servers.

10.7.1 Remote Transaction Restrictions

Remote transaction management has savepoints and nested statement restrictions.

Restrictions on transaction management include the following:

- Savepoints are not propagated to remote servers.
- If nested `BEGIN TRANSACTION` and `COMMIT TRANSACTION` statements are included in a transaction that involves remote servers, only the outermost set of statements is processed. The inconsistent set,

containing the `BEGIN TRANSACTION` and `COMMIT TRANSACTION` statements, is not transmitted to remote servers.

10.7.2 Remote transaction management and restrictions

A multi-phase commit protocol is used for managing transactions that involve remote servers.

However, when more than one remote server is involved in a transaction, there is still a chance that a distributed unit of work will be left in an undetermined state, no recovery process is included.

The general logic for managing a user transaction is as follows:

1. SAP IQ prefaces work to a remote server with a BEGIN TRANSACTION notification.
2. When the transaction is ready to be committed, SAP IQ sends a PREPARE TRANSACTION notification to each remote server that has been part of the transaction. This ensures that the remote server is ready to commit the transaction.
3. If a PREPARE TRANSACTION request fails, all remote servers are instructed to roll back the current transaction.
   If all PREPARE TRANSACTION requests are successful, the server sends a COMMIT TRANSACTION request to each remote server involved with the transaction.

Any statement preceded by BEGIN TRANSACTION can begin a transaction. Other statements are sent to a remote server to be executed as a single, remote unit of work.

Restrictions on transaction management

Restrictions on transaction management are as follows:

- Savepoints are not propagated to remote servers.
- If nested BEGIN TRANSACTION and COMMIT TRANSACTION statements are included in a transaction that involves remote servers, only the outermost set of statements is processed. The innermost set, containing the BEGIN TRANSACTION and COMMIT TRANSACTION statements, is not transmitted to remote servers.

11 Create Procedures and Batches

Procedures and batches enhance the security, efficiency, and standardization of SAP IQ databases.

Procedures store SQL statements in the database for use by all applications. They enhance the security, efficiency, and standardization of databases. User-defined functions are a type of procedure that return a value to the calling environment for use in queries and other SQL statements.

For many purposes, server-side JDBC provides a more flexible way to build logic into the database than SQL stored procedures.

Batches are sets of SQL statements that are submitted to the database server as a group. Many features available in procedures, such as control statements, are also available in batches.

> **i** Note
>
> Use source control software to track changes to source code, and changes to objects created from source (including stored procedures) that you deploy to the database.

In this section:

Procedures [page 246]
Procedures perform one or more specific tasks in the database.

User-Defined Functions [page 256]
User-defined functions are a class of procedures that return a single value to the calling environment.

Batches [page 259]
A batch is a set of SQL statements submitted together and executed as a group, one after the other.

Control statements [page 262]
There are several control statements for logical flow and decision making in the body of a procedure, trigger, or user-defined function, or in a batch.

Structure of Procedures [page 265]
The body of a procedure consists of a compound statement.

Procedure Results [page 268]
Procedures can return either single or multiple rows of data.

Error and warning handling [page 275]
After an application program executes a SQL statement, it can examine a **status code** (or return code) which indicates whether the statement executed successfully or failed and gives the reason for the failure.

Transactions and savepoints in procedures [page 282]
SQL statements in a procedure or trigger are part of the current transaction.

Hiding the contents of a procedure, function, trigger, event, or view [page 283]
Use the SET HIDDEN clause to obscure the contents of a procedure, function, trigger, event, or view.

Statements allowed in procedures, triggers, events, and batches [page 285]
Most SQL statements are acceptable in batches, but there are several exceptions.

11.1 Procedures

Procedures perform one or more specific tasks in the database.

In this section:

Creating Procedures [page 247]
To create an SAP IQ procedure, use the SQL statement `CREATE PROCEDURE`.

Altering Procedures [page 247]
You can modify an existing procedure using either SAP IQ Cockpit or Interactive SQL. You must have the ALTER ANY PROCEDURE system privilege or be the owner of the procedure.

Calling Procedures [page 248]
`CALL` statements invoke procedures. Procedures can be called by an application program or by other procedures.

Deleting Procedures [page 248]
Once you create a procedure, it remains in the database until someone explicitly removes it. Only the owner of the procedure or a user with the DROP ANY PROCEDURE system privilege can drop the procedure from the database.

Privileges to Execute Stored Procedures [page 248]
A procedure is owned by the user who created it; that user can execute it without privilege.

Returning Procedure Results in Parameters [page 248]
Procedures return results to the calling environment in several ways.

Returning Procedure Results in Result Sets [page 249]
In addition to returning results to the calling environment in individual parameters, procedures can return information in result sets. A result set is typically the result of a query.

Displaying Procedure Information [page 249]
Use `sp_iqprocedure` to show information about system and user-defined procedures in a database.

Displaying Procedure Parameter Information [page 250]
Use `sp_procparm` to show information about stored procedure parameters.

Cursors in Procedures [page 250]
Cursors retrieve rows one at a time from a query or stored procedure that has multiple rows in its result set.

Using IQ UTILITIES to Create Stored Procedures [page 253]
The system stored procedures provided in SAP IQ are implemented in SQL.

11.1.1 Creating Procedures

To create an SAP IQ procedure, use the SQL statement `CREATE PROCEDURE`.

Prerequisites

You must have the CREATE PROCEDURE or CREATE ANY PROCEDURE system privilege.

Procedure

Enter a `CREATE PROCEDURE` statement.

This example uses the SAP IQ demo database `iqdemo.db`:

```
CREATE PROCEDURE new_dept(IN id INT,
             IN name CHAR(35),
                IN head_id INT)
BEGIN
        INSERT
                INTO GROUPO.departments(DepartmentID,
                DepartmentName,
                DepartmentHeadID)
                values (id, name, head_id);
END
```

> **i** Note
>
> Use the `AT <location-string>` SQL syntax of `CREATE PROCEDURE` to create a proxy stored procedure.

11.1.2 Altering Procedures

You can modify an existing procedure using either SAP IQ Cockpit or Interactive SQL. You must have the ALTER ANY PROCEDURE system privilege or be the owner of the procedure.

You can also modify procedures using the `ALTER PROCEDURE` statement.

11.1.3 Calling Procedures

`CALL` statements invoke procedures. Procedures can be called by an application program or by other procedures.

11.1.4 Deleting Procedures

Once you create a procedure, it remains in the database until someone explicitly removes it. Only the owner of the procedure or a user with the DROP ANY PROCEDURE system privilege can drop the procedure from the database.

Execute a `DROP` statement specifying a `PROCEDURE` clause to drop the procedure from the database.

11.1.5 Privileges to Execute Stored Procedures

A procedure is owned by the user who created it; that user can execute it without privilege.

> **i** Note
>
> This information does not apply to system procedures. See *SAP IQ SQL Reference* > *System Procedures* for details on granting privilege to run system procedures.

Permission to execute the procedure can be granted to other users using the `GRANT EXECUTE` command. For example, the owner of the procedure `new_dept` allows `another_user` to execute the procedure using:

```
GRANT EXECUTE ON new_dept TO another_user
```

To revoke privilege to execute the procedure, execute:

```
REVOKE EXECUTE ON new_dept FROM another_user
```

11.1.6 Returning Procedure Results in Parameters

Procedures return results to the calling environment in several ways.

- Individual values are returned as OUT or INOUT parameters
- As result sets
- A single result can be returned using a `RETURN` statement

This example uses the SAP IQ demo database `iqdemo.db`:

```
CREATE PROCEDURE SalaryList (IN department_id INT)
RESULT ( "Employee ID" INT, "Salary" NUMERIC(20,3) )
BEGIN
            SELECT EmployeeID, Salary
```

```
            FROM Employees
            WHERE Employees.DepartmentID = department_id;
END
```

11.1.7 Returning Procedure Results in Result Sets

In addition to returning results to the calling environment in individual parameters, procedures can return information in result sets. A result set is typically the result of a query.

If a procedure dynamically creates and then selects the same temporary table within a stored procedure, you must use the `EXECUTE IMMEDIATE WITH RESULT SET ON` syntax to avoid `Column not found` errors.

For example:

```
CREATE PROCEDURE p1 (IN @t varchar(30))
   BEGIN
      EXECUTE IMMEDIATE
      'SELECT * INTO #resultSet FROM ' || @t;
      EXECUTE IMMEDIATE WITH RESULT SET ON
      'SELECT * FROM #resultSet';
   END
```

Related Information

Procedure Results [page 268]

11.1.8 Displaying Procedure Information

Use `sp_iqprocedure` to show information about system and user-defined procedures in a database.

Procedure

In Interactive SQL, run the `sp_iqprocedure` stored procedure.

Displays information about the user-defined procedure `sp_test`:

```
sp_iqprocedure sp_test
```

```
proc_name    proc_owner    proc_defn       replicate     srvid     remarks
sp_test      DBA        create procedure   N             (NULL)    (NULL)
                        DBA.sp_test(in n1
                        integer)
                        begin message'sp_test'end
```

11.1.9 Displaying Procedure Parameter Information

Use `sp_procparm` to show information about stored procedure parameters.

Procedure

In Interactive SQL, run `sp_iqprocparm`.

Display information about the parameters of the system procedure `sp_iqshowcompression`:

```
sp_iqprocparm sp_iqshowcompression, dbo, system
```

```
proc_name             proc_owner  parm_name    parm_type  parm_mode
domain_name  width  scale  default
sp_iqshowcompression  dbo         @owner_name  normal     in
char         128    0      (NULL)
sp_iqshowcompression  dbo         @table_name  normal     in
char         128    0      (NULL)
sp_iqshowcompression  dbo         @column_name normal     in
char         128    0      (NULL)
sp_iqshowcompression  dbo         Column       result     out
char         128    0      (NULL)
sp_iqshowcompression  dbo         Compression  result     out
char         3      0      (NULL)
```

11.1.10 Cursors in Procedures

Cursors retrieve rows one at a time from a query or stored procedure that has multiple rows in its result set.

A cursor is a handle or an identifier for the query or procedure, and for a current position within the result set.

In this section:

Cursor management [page 251]
Managing a cursor is similar to managing a file in a programming language.

Cursor Positioning [page 251]
Cursor positioning is flexible. When a cursor is opened, it is positioned before the first row. You can move the cursor position to an absolute position from the start or the end of the query results, or to a position relative to the current cursor position. The specifics of how you change cursor position, and what operations are possible, are governed by the programming interface.

Cursors on SELECT statements [page 252]
You can use a cursor on a SELECT statement.

11.1.10.1 Cursor management

Managing a cursor is similar to managing a file in a programming language.

The following steps manage cursors:

1. Declare a cursor for a particular SELECT statement or procedure using the DECLARE statement.
2. Open the cursor using the OPEN statement.
3. Use the FETCH statement to retrieve results one row at a time from the cursor.
4. A row not found warning signals the end of the result set.
5. Close the cursor using the CLOSE statement.

By default, cursors are automatically closed at the end of a transaction (on COMMIT or ROLLBACK statements). Cursors opened using the WITH HOLD clause stay open for subsequent transactions until explicitly closed.

11.1.10.2 Cursor Positioning

Cursor positioning is flexible. When a cursor is opened, it is positioned before the first row. You can move the cursor position to an absolute position from the start or the end of the query results, or to a position relative to the current cursor position. The specifics of how you change cursor position, and what operations are possible, are governed by the programming interface.

The number of row positions you can fetch in a cursor is governed by the size of an integer. You can fetch rows numbered up to number 2147483646, which is one less than the value that can be held in an integer. When using negative numbers (rows from the end) you can fetch down to one more than the largest negative value that can be held in an integer.

You can use special positioned update and delete operations to update or delete the row at the current position of the cursor. If the cursor is positioned before the first row or after the last row, an error is returned indicating that there is no corresponding cursor row.

> **i** Note
>
> Inserts and some updates to asensitive cursors can cause problems with cursor positioning. SAP IQ does not put inserted rows at a predictable position within a cursor unless there is an `ORDER BY` clause on the SELECT statement. Sometimes the inserted row does not appear at all until the cursor is closed and opened again. With SAP IQ, this occurs if a work table had to be created to open the cursor.
>
> The `UPDATE` statement may cause a row to move in the cursor. This happens if the cursor has an ORDER BY clause that uses an existing index (a work table is not created). Using `STATIC SCROLL` cursors alleviates these problems but requires more memory and processing.

> **i** Note
>
> SAP IQ treats the `FIRST`, `LAST`, and `ABSOLUTE` options as starting from the beginning of the result set. It treats `RELATIVE` as a negative row count as starting from the current position.

11.1.10.3 Cursors on SELECT statements

You can use a cursor on a SELECT statement.

Based on the same query used in the ListCustomerValue procedure, the example below illustrates features of the stored procedure language.

```
CREATE PROCEDURE TopCustomerValue(
      OUT TopCompany CHAR(36),
      OUT TopValue INT )
BEGIN
   -- 1. Declare the "row not found" exception
   DECLARE err_notfound
      EXCEPTION FOR SQLSTATE '02000';
   -- 2.  Declare variables to hold
   --    each company name and its value
   DECLARE ThisName CHAR(36);
   DECLARE ThisValue INT;
   -- 3.  Declare the cursor ThisCompany
   --     for the query
   DECLARE ThisCompany CURSOR FOR
   SELECT CompanyName,
         CAST( sum( SalesOrderItems.Quantity *
               Products.UnitPrice ) AS INTEGER )
         AS value
   FROM Customers
      INNER JOIN SalesOrders
      INNER JOIN SalesOrderItems
      INNER JOIN Products
   GROUP BY CompanyName;
   -- 4. Initialize the values of TopValue
   SET TopValue = 0;
   -- 5. Open the cursor
   OPEN ThisCompany;
   -- 6. Loop over the rows of the query
   CompanyLoop:
   LOOP
      FETCH NEXT ThisCompany
         INTO ThisName, ThisValue;
      IF SQLSTATE = err_notfound THEN
         LEAVE CompanyLoop;
      END IF;
      IF ThisValue > TopValue THEN
         SET TopCompany = ThisName;
         SET TopValue = ThisValue;
      END IF;
   END LOOP CompanyLoop;
   -- 7. Close the cursor
   CLOSE ThisCompany;
END;
```

Notes

The TopCustomerValue procedure has the following notable features:

- An exception is declared. This exception signals, later in the procedure, when a loop over the results of a query completes.
- Two local variables ThisName and ThisValue are declared to hold the results from each row of the query.
- The cursor ThisCompany is declared. The SELECT statement produces a list of company names and the total value of the orders placed by that company.

- The value of TopValue is set to an initial value of 0, for later use in the loop.
- The ThisCompany cursor opens.
- The LOOP statement loops over each row of the query, placing each company name in turn into the variables ThisName and ThisValue. If ThisValue is greater than the current top value, TopCompany and TopValue are reset to ThisName and ThisValue.
- The cursor closes at the end of the procedure.
- You can also write this procedure without a loop by adding an ORDER BY value DESC clause to the SELECT statement. Then, only the first row of the cursor needs to be fetched.

The LOOP construct in the TopCompanyValue procedure is a standard form, exiting after the last row is processed. You can rewrite this procedure in a more compact form using a FOR loop. The FOR statement combines several aspects of the above procedure into a single statement.

```
CREATE PROCEDURE TopCustomerValue2(
      OUT TopCompany CHAR(36),
      OUT TopValue INT )
BEGIN
   -- 1. Initialize the TopValue variable
   SET TopValue = 0;
   -- 2. Do the For Loop
   FOR CompanyFor AS ThisCompany
      CURSOR FOR
      SELECT CompanyName AS ThisName,
         CAST( sum( SalesOrderItems.Quantity *
               Products.UnitPrice ) AS INTEGER )
         AS ThisValue
      FROM Customers
         INNER JOIN SalesOrders
         INNER JOIN SalesOrderItems
         INNER JOIN Products
      GROUP BY ThisName
   DO
      IF ThisValue > TopValue THEN
         SET TopCompany = ThisName;
         SET TopValue = ThisValue;
      END IF;
   END FOR;
END;
```

11.1.11 Using IQ UTILITIES to Create Stored Procedures

The system stored procedures provided in SAP IQ are implemented in SQL.

Context

You must use the local temporary table and `IQ UTILITIES` statement in exactly the same way as system stored procedures:

All SQL code for procedures is encrypted and compiled into the shared library `libiqscripts16_r.so` file on UNIX and `iqscripts16.dll` file on Windows.

> ⚠ Caution
>
> Failing to use the statements correctly can cause serious problems for your IQ server or database.

To view the stored procedures code, enter `sp_helptext 'owner.procname'` in Interactive SQL.

The syntax for `IQ UTILITIES` is:

```
IQ UTILITIES MAIN INTO <local-temp-table-name> <arguments>
```

You may want to create your own variants of procedures. For example:

- Create a procedure that calls a system stored procedure.
- Create a procedure that is independent of the system stored procedures but performs a similar function.
- Create a procedure that uses the same structure as the system stored procedures but provides additional functionality. For example to display procedure results in graphical form in a front-end tool or browser rather than as text.
- If you choose either of the two previous options, make sure you understand the `IQ UTILITIES` statement and the strict requirements for using it.

In this section:

IQ UTILITIES Command [page 254]
`IQ UTILITIES` is the underlying statement that executes whenever you run most IQ system procedures. In most cases, users are unaware that `IQ UTILITIES` is executing. The only time `IQ UTILITIES` is issued directly by users is to run the IQ buffer cache monitor.

Choosing Procedures to Call [page 255]
You can safely use the `IQ UTILITIES` statement to create your own versions of documented system procedures that report on information in the database.

Numbers Used by IQ UTILITIES [page 255]
Numbers are used as arguments in the `IQ UTILITIES` command. Each number is used in conjunction with a system procedure.

Procedure Testing [page 256]
To maintain the stability of your IQ server and database, test your procedures in a development environment before you run them in a production environment.

11.1.11.1 IQ UTILITIES Command

`IQ UTILITIES` is the underlying statement that executes whenever you run most IQ system procedures. In most cases, users are unaware that `IQ UTILITIES` is executing. The only time `IQ UTILITIES` is issued directly by users is to run the IQ buffer cache monitor.

`IQ UTILITIES` provides a systematic way to collect and report on information maintained in the IQ system tables. There is no general user interface; you can use `IQ UTILITIES` only in the ways that existing system procedures do.

System procedures declare local temporary tables in which to store information. When you execute a system procedure, it in turn executes `IQ UTILITIES` to get the information from the system tables and store it in the

local temporary table. The system procedures may simply report the information from the local temporary table or perform additional processing.

In some system procedures, the `IQ UTILITIES` statement includes a predefined number as one of its arguments. This number performs a specific function, for example, deriving a value from information in the system tables.

11.1.11.2 Choosing Procedures to Call

You can safely use the `IQ UTILITIES` statement to create your own versions of documented system procedures that report on information in the database.

For example, `sp_iqspaceused` displays information about used and available space available in the IQ main and IQ temporary stores. Verify that the procedure you create from a system stored procedure has the correct owner.

Do not create your own versions of system procedures that control IQ operations. Modifying procedures that control IQ operations can lead to serious problems.

11.1.11.3 Numbers Used by IQ UTILITIES

Numbers are used as arguments in the `IQ UTILITIES` command. Each number is used in conjunction with a system procedure.

`IQ UTILITIES` values used in system procedures are:

- 10000 – `sp_iqtransaction`
- 20000 – `sp_iqconnection` and `sp_iqmpxcountdbremote`
- 30000 – `sp_iqspaceused`
- 40000 – `sp_iqspaceinfo`
- 50000 – `sp_iqlocks`
- 60000 – `sp_iqmpxversionfetch`. Do not use.
- 70000 – `sp_iqmpxdumptlvlog`
- 80000 – `sp_iqcontext`
- 100000 – `sp_iqindexfragmentation`
- 110000 – `sp_iqrowdensity`

11.1.11.4 Procedure Testing

To maintain the stability of your IQ server and database, test your procedures in a development environment before you run them in a production environment.

11.2 User-Defined Functions

User-defined functions are a class of procedures that return a single value to the calling environment.

In this section:

Creating User-Defined Functions [page 256]
Use the `CREATE FUNCTION` statement to create user-defined functions.

Calling User-Defined Functions [page 257]
You can use user-defined functions, subject to permissions, in the same places you would use built-in nonaggregate functions.

Dropping a user-defined function (SQL) [page 257]
Drop a user-defined function.

Permissions to Execute User-Defined Functions [page 258]
A user-defined function is owned by the user who created it; that user can execute it without privilege.

Granting the ability to execute a user-defined function (SQL) [page 258]
Grant the ability to execute a user-defined function by granting the EXECUTE object-level privilege.

11.2.1 Creating User-Defined Functions

Use the `CREATE FUNCTION` statement to create user-defined functions.

To create user-defined functions, you need the following:

- You must have the CREATE PROCEDURE system privilege to create functions owned by you.
- You must have the CREATE ANY PROCEDURE or CREATE ANY OBJECT system privilege to create functions owned by others.
- If the procedure contains an external reference, you must have the CREATE EXTERNAL REFERENCE system privilege, in addition to the above system privileges.

See *CREATE FUNCTION Statement* in SAP IQ SQL Reference for syntax and usage details. To create a user-defined function in SAP IQ Cockpit, see the *SAP IQ Cockpit* guide.

11.2.2 Calling User-Defined Functions

You can use user-defined functions, subject to permissions, in the same places you would use built-in nonaggregate functions.

This Interactive SQL statement returns a full name from two columns containing a first and last name:

```
SELECT fullname (GivenName, LastName)
FROM Employees;
```

fullname (Employees.GivenName,Employees.LastName)
Fran Whitney
Matthew Cobb
Philip Chin
...

The following statement returns a full name from a supplied first and last name:

```
SELECT fullname ('Jane', 'Smith');
```

fullname ('Jane','Smith')
Jane Smith

Any user who has been granted Execute permissions for the function can use the `<fullname>` function.

11.2.3 Dropping a user-defined function (SQL)

Drop a user-defined function.

Prerequisites

You must be the owner of the user-defined function or have one of the following system privileges:

- DROP ANY PROCEDURE
- DROP ANY OBJECT

Procedure

1. Connect to the database.
2. Execute a DROP FUNCTION statement similar to the following:

   ```
   DROP FUNCTION <function-name>;
   ```

Results

The user-defined function is dropped.

Example

The following statement removes the function FullName from the database:

```
DROP FUNCTION FullName;
```

11.2.4 Permissions to Execute User-Defined Functions

A user-defined function is owned by the user who created it; that user can execute it without privilege.

The owner of a user-defined function can grant privileges to other users with the `GRANT EXECUTE` command.

For example, the creator of the function `fullname` allows `another_user` to use `fullname` the function using:

```
GRANT EXECUTE ON fullname TO another_user
```

This statement revokes privilege to use the function:

```
REVOKE EXECUTE ON fullname FROM another_user
```

See *Granting the Privilege to Run a Function or Procedure* in *SAP IQ Administration: User Management and Security*.

11.2.5 Granting the ability to execute a user-defined function (SQL)

Grant the ability to execute a user-defined function by granting the EXECUTE object-level privilege.

Prerequisites

You must be the owner of the user-defined function, or have EXECUTE privilege with administrative rights on the function.

Ownership of a user-defined function belongs to the user who created it, and no privilege is required for that user to execute it.

Context

You have created a function and you want other user to be able to use it.

Procedure

1. Connect to the database.
2. Execute a GRANT EXECUTE statement similar to the following:

```
GRANT EXECUTE ON <function-name> TO <user-id>;
```

Results

The grantee can now execute the procedure.

Example

For example, the creator of the Nationality function could allow another user to use Nationality with the statement:

```
GRANT EXECUTE ON Nationality TO BobS;
```

11.3 Batches

A batch is a set of SQL statements submitted together and executed as a group, one after the other.

The control statements used in procedures (CASE, IF, LOOP, and so on) can also be used in batches. If the batch consists of a compound statement enclosed in a BEGIN/END, then it can also contain host variables, local declarations for variables, cursors, temporary tables and exceptions. Host variable references are permitted within batches with the following restrictions:

- only one statement in the batch can refer to host variables
- the statement which uses host variables cannot be preceded by a statement which returns a result set

Use of BEGIN/END is recommended to clearly indicate when a batch is being used.

Statements within the batch may be delimited with semicolons, in which case the batch is conforming to the Watcom SQL dialect. A multi-statement batch that does not use semicolons to delimit statements conforms to the Transact-SQL dialect. The dialect of the batch determines which statements are permitted within the batch, and also determines how errors within the batch are handled.

In many ways, batches are similar to stored procedures; however, there are some differences:

- batches do not have names
- batches do not accept parameters
- batches are not stored persistently in the database
- batches cannot be shared by different connections

A simple batch consists of a set of SQL statements with no delimiters followed by a separate line with just the word go on it. The following example creates an Eastern Sales department and transfers all sales reps from Massachusetts to that department. It is an example of a Transact-SQL batch.

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 220, 'Eastern Sales' )
UPDATE Employees
SET DepartmentID = 220
WHERE DepartmentID = 200
AND State = 'MA'
COMMIT
go
```

The word go is recognized by Interactive SQL and causes it to send the previous statements as a single batch to the server.

The following example, while similar in appearance, is handled quite differently by Interactive SQL. This example does not use the Transact-SQL dialect. Each statement is delimited by a semicolon. Interactive SQL sends each semicolon-delimited statement separately to the server. It is not treated as a batch.

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 220, 'Eastern Sales' );
UPDATE Employees
SET DepartmentID = 220
WHERE DepartmentID = 200
AND State = 'MA';
COMMIT;
```

To have Interactive SQL treat it as a batch, it can be changed into a compound statement using `BEGIN ... END`. The following is a revised version of the previous example. The three statements in the compound statement are sent as a batch to the server.

```
BEGIN
  INSERT
  INTO Departments ( DepartmentID, DepartmentName )
  VALUES ( 220, 'Eastern Sales' );
  UPDATE Employees
  SET DepartmentID = 220
  WHERE DepartmentID = 200
  AND State = 'MA';
  COMMIT;
END
```

In this particular example, it makes no difference to the end result whether a batch or individual statements are executed by the server. There are situations, though, where it can make a difference. Consider the following example.

```
DECLARE @CurrentID INTEGER;
SET @CurrentID = 207;
SELECT Surname FROM Employees
```

```
  WHERE EmployeeID=@CurrentID;
```

If you execute this example using Interactive SQL, the database server returns an error indicating that the variable cannot be found. This happens because Interactive SQL sends three separate statements to the server. They are not executed as a batch. As you have already seen, the remedy is to use a compound statement to force Interactive SQL to send these statements as a batch to the server. The following example accomplishes this.

```
BEGIN
  DECLARE @CurrentID INTEGER;
  SET @CurrentID = 207;
  SELECT Surname FROM Employees
    WHERE EmployeeID=@CurrentID;
END
```

Putting a BEGIN and END around a set of statements forces Interactive SQL to treat them as a batch.

The IF statement is another example of a compound statement. Interactive SQL sends the following statements as a single batch to the server.

```
IF EXISTS(   SELECT *
             FROM SYSTAB
             WHERE table_name='Employees' )
THEN
   SELECT   Surname AS LastName,
            GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
ELSE
   MESSAGE 'The Employees table does not exist'
   TO CLIENT;
END IF
```

This situation does not arise when using other techniques to prepare and execute SQL statements. For example, an application that uses ODBC can prepare and execute a series of semicolon-separated statements as a batch.

Care must be exercised when mixing Interactive SQL statements with SQL statements intended for the server. The following is an example of how mixing Interactive SQL statements and SQL statements can be an issue. In this example, since the Interactive SQL OUTPUT statement is embedded in the compound statement, it is sent along with all the other statements to the server as a batch, and results in a syntax error.

```
IF EXISTS(   SELECT *
             FROM SYSTAB
             WHERE table_name='Employees' )
THEN
   SELECT   Surname AS LastName,
            GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
   OUTPUT TO 'c:\\temp\\query.txt';
ELSE
   MESSAGE 'The Employees table does not exist'
   TO CLIENT;
END IF
```

The correct placement of the OUTPUT statement is shown below.

```
IF EXISTS(   SELECT *
             FROM SYSTAB
             WHERE table_name='Employees' )
THEN
   SELECT    Surname AS LastName,
             GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
ELSE
   MESSAGE 'The Employees table does not exist'
   TO CLIENT;
END IF;
OUTPUT TO 'c:\\temp\\query.txt';
```

11.4 Control statements

There are several control statements for logical flow and decision making in the body of a procedure, trigger, or user-defined function, or in a batch.

Available control statements include:

Control statement	Syntax

Compound statements	`BEGIN [ ATOMIC ]`
	`   <Statement-list>`
	`END`

Conditional execution: IF	`IF <condition> THEN`
	`   <Statement-list>`
	`ELSEIF <condition> THEN`
	`   <Statement-list>`
	`ELSE`
	`   <Statement-list>`
	`END IF`

Conditional execution: CASE	`CASE <expression>`
	`WHEN <value> THEN`
	`   <Statement-list>`
	`WHEN <value> THEN`
	`   <Statement-list>`
	`ELSE`
	`   <Statement-list>`
	`END CASE`

Repetition: WHILE, LOOP	`WHILE <condition> LOOP`
	`   <Statement-list>`
	`END LOOP`

Control statement	Syntax

Repetition: FOR cursor loop	`FOR <loop-name>`
	`  AS <cursor-name> CURSOR FOR`
	`  <select-statement>`
	`DO`
	`  <Statement-list>`
	`END FOR`

Break: LEAVE	`LEAVE <label>`
CALL	`CALL <procname>( <arg>, ... )`

In this section:

Compound statements [page 263]
The body of a procedure or trigger is a **compound statement**.

Declarations in compound statements [page 263]
Local declarations in a compound statement immediately follow the BEGIN keyword.

Atomic compound statements [page 264]
An **atomic** statement is a statement that is executed completely or not at all.

11.4.1 Compound statements

The body of a procedure or trigger is a **compound statement**.

A compound statement starts with the keyword BEGIN and concludes with the keyword END. Compound statements can also be used in batches. Compound statements can be nested, and combined with other control statements to define execution flow in procedures and triggers or in batches.

A compound statement allows a set of SQL statements to be grouped together and treated as a unit. Delimit SQL statements within a compound statement with semicolons.

11.4.2 Declarations in compound statements

Local declarations in a compound statement immediately follow the BEGIN keyword.

These local declarations exist only within the compound statement. Within a compound statement you can declare:

- Variables
- Cursors
- Temporary tables
- Exceptions (error identifiers)

Local declarations can be referenced by any statement in that compound statement, or in any compound statement nested within it. Local declarations are not visible to other procedures called from the compound statement.

11.4.3 Atomic compound statements

An **atomic** statement is a statement that is executed completely or not at all.

For example, an UPDATE statement that updates thousands of rows might encounter an error after updating many rows. If the statement does not complete, all changed rows revert back to their original state. The UPDATE statement is atomic.

All non-compound SQL statements are atomic. You can make a compound statement atomic by adding the keyword ATOMIC after the BEGIN keyword.

```
BEGIN ATOMIC
   UPDATE Employees
   SET ManagerID = 501
   WHERE EmployeeID = 467;
    UPDATE Employees
   SET BirthDate = 'bad_data';
END
```

In this example, the two update statements are part of an atomic compound statement. They must either succeed or fail as one. The first update statement would succeed. The second one causes a data conversion error since the value being assigned to the BirthDate column cannot be converted to a date.

The atomic compound statement fails and the effect of both UPDATE statements is undone. Even if the currently executing transaction is eventually committed, neither statement in the atomic compound statement takes effect.

If an atomic compound statement succeeds, the changes made within the compound statement take effect only if the currently executing transaction is committed. In the case when an atomic compound statement succeeds but the transaction in which it occurs gets rolled back, the atomic compound statement also gets rolled back. A savepoint is established at the start of the atomic compound statement. Any errors within the statement result in a rollback to that savepoint.

When an atomic compound statement is executed in autocommit (unchained) mode, the commit mode changes to manual (chained) until statement execution is complete. In manual mode, DML statements executed within the atomic compound statement do not cause an immediate COMMIT or ROLLBACK. If the atomic compound statement completes successfully, a COMMIT statement is executed; otherwise, a ROLLBACK statement is executed.

You cannot use COMMIT and ROLLBACK and some ROLLBACK TO SAVEPOINT statements within an atomic compound statement.

11.5 Structure of Procedures

The body of a procedure consists of a compound statement.

A compound statement consists of a `BEGIN` and an `END`, enclosing a set of SQL statements. Semicolons delimit each statement.

In this section:

SQL Statements Allowed in Procedures [page 265]
You can use almost all SQL statements within procedures.

Parameter declaration for procedures [page 266]
Procedure parameters appear as a list in the CREATE PROCEDURE statement.

Ways to pass parameters to procedures [page 266]
You can take advantage of default values of stored procedure parameters with either of two forms of the CALL statement.

How to pass parameters to functions [page 267]
User-defined functions are not invoked with the CALL statement, but are used in the same manner that built-in functions are.

11.5.1 SQL Statements Allowed in Procedures

You can use almost all SQL statements within procedures.

- `SELECT`, `UPDATE`, `DELETE`, `INSERT`, and `SET VARIABLE`
- The `CALL` statement to execute other procedures
- Control statements
- Cursor statements
- Exception handling statements
- The `EXECUTE IMMEDIATE` statement

Some SQL statements you cannot use within procedures:

- `CONNECT` statement
- `DISCONNECT` statement

You can use `COMMIT`, `ROLLBACK`, and `SAVEPOINT` statements within procedures with certain restrictions.

See each statement in *SAP IQ SQL Reference* > *SQL Statements*.

11.5.2 Parameter declaration for procedures

Procedure parameters appear as a list in the CREATE PROCEDURE statement.

Parameter names must conform to the rules for other database identifiers such as column names. They must have valid data types, and can be prefixed with one of the keywords IN, OUT or INOUT. By default, parameters are INOUT parameters. These keywords have the following meanings:

IN

The argument is an expression that provides a value to the procedure.

OUT

The argument is a variable that could be given a value by the procedure.

INOUT

The argument is a variable that provides a value to the procedure, and could be given a new value by the procedure.

You can assign default values to procedure parameters in the CREATE PROCEDURE statement. The default value must be a constant, which may be NULL. For example, the following procedure uses the NULL default for an IN parameter to avoid executing a query that would have no meaning:

```
CREATE PROCEDURE CustomerProducts( IN customer_ID INTEGER DEFAULT NULL )
RESULT ( product_ID INTEGER,
         quantity_ordered INTEGER )
BEGIN
   IF customer_ID IS NULL THEN
      RETURN;
   ELSE
      SELECT Products.ID, sum( SalesOrderItems.Quantity )
      FROM   Products, SalesOrderItems, SalesOrders
      WHERE  SalesOrders.CustomerID = customer_ID
        AND  SalesOrders.ID = SalesOrderItems.ID
        AND  SalesOrderItems.ProductID = Products.ID
      GROUP BY Products.ID;
   END IF;
END;
```

The following statement assigns the DEFAULT NULL, and the procedure performs a RETURN operation instead of executing the query.

```
CALL CustomerProducts();
```

11.5.3 Ways to pass parameters to procedures

You can take advantage of default values of stored procedure parameters with either of two forms of the CALL statement.

If the optional parameters are at the end of the argument list in the CREATE PROCEDURE statement, they may be omitted from the CALL statement. As an example, consider a procedure with three INOUT parameters:

```
CREATE PROCEDURE SampleProcedure(
       INOUT var1 INT DEFAULT 1,
                      INOUT var2 int DEFAULT 2,
```

```
                      INOUT var3 int DEFAULT 3 )
...
```

This next example assumes that the calling environment has set up three connection-scope variables to hold the values passed to the procedures.

```
CREATE VARIABLE V1 INT;
CREATE VARIABLE V2 INT;
CREATE VARIABLE V3 INT;
```

The procedure SampleProcedure may be called supplying only the first parameter as follows, in which case the default values are used for *var2* and *var3*.

```
CALL SampleProcedure( V1 );
```

The procedure can also be called by providing only the second parameter by using the DEFAULT value for the first parameter, as follows:

```
CALL SampleProcedure( DEFAULT, V2 );
```

A more flexible method of calling procedures with optional arguments is to pass the parameters by name. The SampleProcedure procedure may be called as follows:

```
CALL SampleProcedure( var1 = V1, var3 = V3 );
```

or as follows:

```
CALL SampleProcedure( var3 = V3, var1 = V1 );
```

> **i** Note
>
> Database-scope variables cannot be used for INOUT and OUT parameters when calling a procedure. They can be used for IN parameters, however.

11.5.4 How to pass parameters to functions

User-defined functions are not invoked with the CALL statement, but are used in the same manner that built-in functions are.

For example, the following example uses the FullName function to retrieve the names of employees:

Example

In Interactive SQL, execute the following query:

```
SELECT FullName( GivenName, Surname ) AS Name
   FROM Employees;
```

The following results appear:

Name
Fran Whitney
Matthew Cobb
Philip Chin
Julie Jordan
...

Notes

- Default parameters can be used in calling functions. However, parameters cannot be passed to functions by name.
- Parameters are passed by value, not by reference. Even if the function changes the value of the parameter, this change is not returned to the calling environment.
- Output parameters cannot be used in user-defined functions.
- User-defined functions cannot return result sets.

11.6 Procedure Results

Procedures can return either single or multiple rows of data.

Results consisting of a single row of data can be passed back as arguments to the procedure. Results consisting of multiple rows of data are passed back as result sets. Procedures can also return a single value given in the `RETURN` statement.

In this section:

Returning a value using the RETURN statement [page 269]
The RETURN statement returns a single integer value to the calling environment, causing an immediate exit from the procedure.

Ways to return results as procedure parameters [page 269]
Procedures can return results to the calling environment in the parameters to the procedure.

Information returned in result sets from procedures [page 271]
Procedures can return information in result sets.

Returning multiple result sets [page 273]
Use Interactive SQL to return more than one result set from a procedure.

Variable result sets for procedures [page 274]
Omitting the RESULT clause allows you to write procedures that return different result sets, with different numbers or types of columns, depending on how they are executed.

11.6.1 Returning a value using the RETURN statement

The RETURN statement returns a single integer value to the calling environment, causing an immediate exit from the procedure.

Procedure

1. Execute the following statement:

   ```
   RETURN <expression>
   ```

2. The value of the supplied expression is returned to the calling environment. Use an extension of the CALL statement to save the return value in a variable:

   ```
   CREATE VARIABLE returnval INTEGER;
   returnval = CALL <variable/procedure-name>? myproc();
   ```

Results

A value is returned and saved as a variable.

11.6.2 Ways to return results as procedure parameters

Procedures can return results to the calling environment in the parameters to the procedure.

Within a procedure, parameters and variables can be assigned values using:

- the SET statement
  The following procedure returns a value in an OUT parameter assigned using a SET statement. You must have the CREATE PROCEDURE system privilege to execute the following statement:

  ```
  CREATE PROCEDURE greater(
     IN a INT,
     IN b INT,
     OUT c INT )
  BEGIN
     IF a > b THEN
        SET c = a;
     ELSE
        SET c = b;
     END IF ;
  END;
  ```

- a SELECT statement with an INTO clause
  A single-row query retrieves at most one row from the database. This type of query uses a SELECT statement with an INTO clause. The INTO clause follows the SELECT list and precedes the FROM clause. It contains a list of variables to receive the value for each SELECT list item. There must be the same number of variables as there are SELECT list items.

When a SELECT statement executes, the database server retrieves the results of the SELECT statement and places the results in the variables. If the query results contain more than one row, the database server returns an error. For queries returning more than one row, you must use cursors.
If the query results in no rows being selected, the variables are not updated, and a warning is returned.
You must have the appropriate SELECT privileges on the object to execute a SELECT statement.

Example

Example 1: Create a procedure and select its results using a SELECT...INTO statement

1. Start Interactive SQL and connect to the SAP IQ sample database. You must have the CREATE PROCEDURE system privilege and either SELECT privilege on the Employees table or the SELECT ANY TABLE system privilege.
2. In the **SQL Statements** pane, execute the following statement to create a procedure (AverageSalary) that returns the average salary of employees as an OUT parameter:

```
CREATE PROCEDURE AverageSalary( OUT average_salary NUMERIC(20,3) )
BEGIN
   SELECT AVG( Salary )
   INTO average_salary
   FROM GROUPO.Employees;
END;
```

3. Create a variable to hold the procedure output. In this case, the output variable is numeric, with three decimal places.

```
CREATE VARIABLE Average NUMERIC(20,3);
```

4. Call the procedure using the created variable to hold the result:

```
CALL AverageSalary( Average );
```

5. If the procedure was created and run properly, the Interactive SQL **History** tab does not display any errors.
6. To inspect the value of the variable, execute the following statement:

```
SELECT Average;
```

7. Look at the value of the output variable Average. The **Results** tab in the **Results** pane displays the value 49988.623 for this variable, the average employee salary.

Example 2: Returning the results of a single-row SELECT statement

1. Start Interactive SQL and connect to the SAP IQ sample database. You must have the CREATE PROCEDURE system privilege and either SELECT privilege on the Customers table or the SELECT ANY TABLE system privilege.
2. Execute the following statement to return the number of orders placed by a given customer:

```
CREATE PROCEDURE OrderCount(
   IN customer_ID INT,
   OUT Orders INT )
BEGIN
   SELECT COUNT(SalesOrders.ID)
      INTO Orders
   FROM GROUPO.Customers
      KEY LEFT OUTER JOIN SalesOrders
   WHERE Customers.ID = customer_ID;
```

```
END;
```

3. Test this procedure using the following statements, which show the number of orders placed by the customer with ID 102:

```
CREATE VARIABLE orders INT;
CALL OrderCount ( 102, orders );
SELECT orders;
```

Notes for Example 2

- The customer_ID parameter is declared as an IN parameter. This parameter holds the customer ID passed in to the procedure.
- The Orders parameter is declared as an OUT parameter. It holds the value of the orders variable returned to the calling environment.
- No DECLARE statement is necessary for the Orders variable as it is declared in the procedure argument list.
- The SELECT statement returns a single row and places it into the variable Orders.

11.6.3 Information returned in result sets from procedures

Procedures can return information in result sets.

The number of variables in the RESULT clause must match the number of the SELECT list items. Automatic data type conversion is performed where possible if data types do not match. The names of the SELECT list items do not have to match those in the RESULT clause.

The RESULT clause is part of the CREATE PROCEDURE statement, and does not have a statement delimiter.

To modify procedure result sets on a view, the user must have the appropriate privileges on the underlying table.

If a stored procedure or user-defined function returns a result, then it cannot also support output parameters or return values.

By default, Interactive SQL only displays the first result set. To allow a procedure to return more than one row of results in Interactive SQL, set the **Show Multiple Result Sets** option on the **Results** tab of the **Options** window.

Example

Example 1

The following procedure returns a list of customers who have placed orders, together with the total value of the orders placed.

Execute the following statement in Interactive SQL:

```
CREATE PROCEDURE ListCustomerValue()
RESULT ( "Company" CHAR(36), "Value" INT )
BEGIN
   SELECT CompanyName,
```

```
      CAST( SUM(  SalesOrderItems.Quantity *
                  Products.UnitPrice )
                  AS INTEGER ) AS value
   FROM Customers
      INNER JOIN SalesOrders
      INNER JOIN SalesOrderItems
      INNER JOIN Products
   GROUP BY CompanyName
   ORDER BY value DESC;
END;
```

Executing `CALL ListCustomerValue ( );` returns the following result set:

Company	Value
The Hat Company	5016
The Igloo	3564
The Ultimate	3348
North Land Trading	3144
Molly's	2808
...	...

Example 2

The following procedure returns a result set containing the salary for each employee in a given department. Execute the following statement in Interactive SQL:

```
CREATE PROCEDURE SalaryList( IN department_id INT )
RESULT ( "Employee ID" INT, Salary NUMERIC(20,3) )
BEGIN
   SELECT EmployeeID, Salary
    FROM Employees
    WHERE Employees.DepartmentID = department_id;
END;
```

The names in the RESULT clause are matched to the results of the query and used as column headings in the displayed results.

To list the salaries of employees in the R & D department (department ID 100), execute the following statement:

```
CALL SalaryList( 100 );
```

The following result set appears in the **Results** pane:

Employee ID	Salary
102	45700.000
105	62000.000
160	57490.000
243	72995.000
...	...

11.6.4 Returning multiple result sets

Use Interactive SQL to return more than one result set from a procedure.

Context

By default, Interactive SQL does not show multiple result sets.

Procedure

1. In Interactive SQL, connect to the database.
2. Click **Tools** > **Options**.
3. Click *SAP IQ*.
4. On the **Results** tab, click **Show All Result Sets**.
5. Click **OK**.

Results

After you enable this option, Interactive SQL shows multiple result sets. The setting takes effect immediately and remains in effect for future sessions until it is disabled.

Example

The following procedure lists the names of all employees, customers, and contacts listed in the database:

```
CREATE PROCEDURE ListPeople()
RESULT ( Surname CHAR(36), GivenName CHAR(36) )
BEGIN
   SELECT Surname, GivenName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
END;
```

Next Steps

If a RESULT clause is included in a procedure definition, the result sets must be compatible: they must have the same number of items in the SELECT lists, and the data types must all be of types that can be automatically converted to the data types listed in the RESULT clause.

If the RESULT clause is omitted, a procedure can return result sets that vary in the number and type of columns that are returned.

11.6.5 Variable result sets for procedures

Omitting the RESULT clause allows you to write procedures that return different result sets, with different numbers or types of columns, depending on how they are executed.

The RESULT clause is optional in procedures. If you do not use the variable result sets feature, use a RESULT clause for performance reasons.

For example, the following procedure returns two columns if the input variable is Y, but only one column otherwise:

```
CREATE PROCEDURE Names( IN formal char(1) )
BEGIN
   IF formal = 'y' THEN
      SELECT Surname, GivenName
      FROM Employees
   ELSE
      SELECT GivenName
      FROM Employees
   END IF
END;
```

The use of variable result sets in procedures is subject to some limitations, depending on the interface used by the client application.

Embedded SQL

To get the proper shape of the result set, you must DESCRIBE the procedure call after the cursor for the result set is opened, but before any rows are returned.

When you create a procedure without a RESULT clause and the procedure returns a variable result set, a DESCRIBE of a SELECT statement that references the procedure may fail. To prevent the failure of the DESCRIBE, it is recommended that you include a WITH clause in the FROM clause of the SELECT statement. Alternately, you could use the WITH VARIABLE RESULT clause in the DESCRIBE statement. The WITH VARIABLE RESULT clause can be used to determine if the procedure call should be described following each OPEN statement.

ODBC

Variable result set procedures can be used by ODBC applications. The SAP IQ ODBC driver performs the proper description of the variable result sets.

Open Client applications

Open Client applications can use variable result set procedures. SAP IQ performs the proper description of the variable result sets.

11.7 Error and warning handling

After an application program executes a SQL statement, it can examine a **status code** (or return code) which indicates whether the statement executed successfully or failed and gives the reason for the failure.

You can use the same mechanism to indicate the success or failure of a CALL statement to a procedure.

Error reporting uses either the SQLCODE or SQLSTATE status descriptions.

Whenever a SQL statement executes, a value appears in special procedure variables called SQLSTATE and SQLCODE. The special value indicates whether there were any unusual conditions encountered when the statement was executed. You can check the value of SQLSTATE or SQLCODE in an IF statement following a SQL statement, and take actions depending on whether the statement succeeded or failed.

For example, the SQLSTATE variable can be used to indicate if a row is successfully fetched. The TopCustomerValue procedure used the SQLSTATE test to detect that all rows of a SELECT statement had been processed.

In this section:

Default handling of errors [page 275]
If you have no error handling built in to a procedure, the database server will handle errors that occur during the procedure execution using its default settings.

Error handling with ON EXCEPTION RESUME [page 277]
If the ON EXCEPTION RESUME clause appears in the CREATE PROCEDURE statement, the procedure checks the following statement when an error occurs.

Default handling of warnings [page 278]
Errors are handled differently than warnings.

Exception handlers [page 279]
You can intercept certain types of errors and handle them within a procedure or trigger, rather than pass the error back to the calling environment. This is done through the use of an **exception handler**.

Exception handling and nested compound statements [page 280]
The code following a statement that causes an error executes only if an ON EXCEPTION RESUME clause appears in a procedure definition.

11.7.1 Default handling of errors

If you have no error handling built in to a procedure, the database server will handle errors that occur during the procedure execution using its default settings.

For different behavior, you can use exception handlers.

Warnings are handled in a slightly different manner from errors.

There are two ways of handling errors without using explicit error handling:

Default error handling

The procedure or trigger fails and returns an error code to the calling environment.

ON EXCEPTION RESUME

If the ON EXCEPTION RESUME clause appears in the CREATE PROCEDURE statement, the procedure carries on executing after an error, resuming at the statement following the one causing the error.

The precise behavior for procedures that use ON EXCEPTION RESUME is dictated by the on_tsql_error option setting.

Default error handling

Generally, if a SQL statement in a procedure or trigger fails, the procedure or trigger stops executing and control returns to the application program with an appropriate setting for the SQLSTATE and SQLCODE values. This is true even if the error occurred in a procedure or trigger invoked directly or indirectly from the first one. For triggers the operation causing the trigger is also undone and the error is returned to the application.

The following demonstration procedures show what happens when an application calls the procedure OuterProc, and OuterProc in turn calls the procedure InnerProc, which then encounters an error.

```
CREATE OR REPLACE PROCEDURE OuterProc()
BEGIN
   MESSAGE 'Hello from OuterProc' TO CLIENT;
   CALL InnerProc();
   MESSAGE 'SQLSTATE set to ', SQLSTATE,' in OuterProc' TO CLIENT;
END;
CREATE OR REPLACE PROCEDURE InnerProc()
BEGIN
      DECLARE column_not_found EXCEPTION FOR SQLSTATE '52003';
      MESSAGE 'Hello from InnerProc' TO CLIENT;
      SIGNAL column_not_found;
      MESSAGE 'SQLSTATE set to ', SQLSTATE, ' in InnerProc' TO CLIENT;
END;
CALL OuterProc();
```

The Interactive SQL **History** tab displays the following:

```
Hello from OuterProc
Hello from InnerProc
```

The DECLARE statement in InnerProc declares a symbolic name for one of the predefined SQLSTATE values associated with error conditions already known to the server.

When executed, the MESSAGE ... TO CLIENT statement sends a message to the Interactive SQL **History** tab.

The SIGNAL statement generates an error condition from within the InnerProc procedure.

None of the statements following the SIGNAL statement in InnerProc execute: InnerProc immediately passes control back to the calling environment, which in this case is the procedure OuterProc. None of the statements following the CALL statement in OuterProc execute. The error condition returns to the calling environment to be handled there. For example, Interactive SQL handles the error by displaying a message window describing the error.

The TRACEBACK function provides a compressed list of the statements that were executing when the error occurred. You can use the SA_SPLIT_LIST system procedure to break up the result from the TRACEBACK function as follows:

```
SELECT * FROM SA_SPLIT_LIST( TRACEBACK(), '\n' )
```

11.7.2 Error handling with ON EXCEPTION RESUME

If the ON EXCEPTION RESUME clause appears in the CREATE PROCEDURE statement, the procedure checks the following statement when an error occurs.

If the statement handles the error, then the procedure continues executing, resuming at the statement after the one causing the error. It does not return control to the calling environment when an error occurred.

The behavior for procedures that use ON EXCEPTION RESUME can be modified by the on_tsql_error option setting.

Error-handling statements include the following:

- IF
- SELECT @variable =
- CASE
- LOOP
- LEAVE
- CONTINUE
- CALL
- EXECUTE
- SIGNAL
- RESIGNAL
- DECLARE
- SET VARIABLE

The following demonstration procedures show what happens when an application calls the procedure OuterProc; and OuterProc in turn calls the procedure InnerProc, which then encounters an error. These demonstration procedures are based on those used earlier:

```
CREATE OR REPLACE PROCEDURE OuterProc()
ON EXCEPTION RESUME
BEGIN
   DECLARE res CHAR(5);
   MESSAGE 'Hello from OuterProc' TO CLIENT;
   CALL InnerProc();
   SET res = SQLSTATE;
   IF res = '52003' THEN
      MESSAGE 'SQLSTATE set to ', res, ' in OuterProc' TO CLIENT;
   END IF
END;
CREATE OR REPLACE PROCEDURE InnerProc()
ON EXCEPTION RESUME
BEGIN
   DECLARE column_not_found EXCEPTION FOR SQLSTATE '52003';
   MESSAGE 'Hello from InnerProc' TO CLIENT;
   SIGNAL column_not_found;
   MESSAGE 'SQLSTATE set to ', SQLSTATE, ' in InnerProc' TO CLIENT;
END;
CALL OuterProc();
```

The Interactive SQL **History** tab then displays the following:

```
Hello from OuterProc
Hello from InnerProc
SQLSTATE set to 52003 in OuterProc
```

The execution path taken is as follows:

1. OuterProc executes and calls InnerProc.
2. In InnerProc, the SIGNAL statement signals an error.
3. The MESSAGE statement is not an error-handling statement, so control is passed back to OuterProc and the message is not displayed.
4. In OuterProc, the statement following the error assigns the SQLSTATE value to the variable named 'res'. This is an error-handling statement, and so execution continues and the OuterProc message appears.

11.7.3 Default handling of warnings

Errors are handled differently than warnings.

While the default action for errors is to set a value for the SQLSTATE and SQLCODE variables, and return control to the calling environment in the event of an error, the default action for warnings is to set the SQLSTATE and SQLCODE values and continue execution of the procedure.

The following demonstration procedures illustrate default handling of warnings.

In this case, the SIGNAL statement generates a condition indicating that the row cannot be found. This is a warning rather than an error.

```
CREATE OR REPLACE PROCEDURE OuterProc()
BEGIN
   MESSAGE 'Hello from OuterProc' TO CLIENT;
   CALL InnerProc();
   MESSAGE 'SQLSTATE set to ', SQLSTATE,' in OuterProc' TO CLIENT;
END;
CREATE OR REPLACE PROCEDURE InnerProc()
BEGIN
   DECLARE row_not_found EXCEPTION FOR SQLSTATE '02000';
   MESSAGE 'Hello from InnerProc' TO CLIENT;
   SIGNAL row_not_found;
   MESSAGE 'SQLSTATE set to ', SQLSTATE, ' in InnerProc' TO CLIENT;
END;
CALL OuterProc();
```

The Interactive SQL **History** tab then displays the following:

```
Hello from OuterProc
Hello from InnerProc
SQLSTATE set to 02000 in InnerProc
SQLSTATE set to 00000 in OuterProc
```

The procedures both continued executing after the warning was generated, with SQLSTATE set by the warning (02000).

Execution of the second MESSAGE statement in InnerProc resets the warning. Successful execution of any SQL statement resets SQLSTATE to 00000 and SQLCODE to 0. If a procedure needs to save the error status, it must do an assignment of the value immediately after execution of the statement, which caused the error or warning.

11.7.4 Exception handlers

You can intercept certain types of errors and handle them within a procedure or trigger, rather than pass the error back to the calling environment. This is done through the use of an **exception handler**.

You define an exception handler with the EXCEPTION part of a compound statement.

Whenever an error occurs in the compound statement, the exception handler executes. Unlike errors, warnings do not cause exception handling code to be executed. Exception handling code also executes if an error appears in a nested compound statement or in a procedure or trigger invoked anywhere within the compound statement.

An exception handler for the interrupt error SQL_INTERRUPT, SQLSTATE 57014 should only contain non-interruptible statements such as ROLLBACK and ROLLBACK TO SAVEPOINT. If the exception handler contains interruptible statements that are invoked when the connection is interrupted, the database server stops the exception handler at the first interruptible statement and returns the interrupt error.

An exception handler can use the SQLSTATE or SQLCODE special values to determine why a statement failed. Alternatively, the ERRORMSG function can be used without an argument to return the error condition associated with a SQLSTATE. Only the first statement in each WHEN clause can specify this information and the statement cannot be a compound statement.

In this example, an exception handler in the InnerProc procedure handles a column not found error. For demonstration purposes, the error is generated artificially using the SIGNAL statement.

An exception handler is also included in the OuterProc procedure.

```
CREATE OR REPLACE PROCEDURE OuterProc()
BEGIN
   MESSAGE 'Hello from OuterProc' TO CLIENT;
   CALL InnerProc();
   MESSAGE 'SQLSTATE set to ', SQLSTATE,' in OuterProc (no exception)' TO CLIENT;
   EXCEPTION
      WHEN OTHERS THEN
        MESSAGE 'SQLSTATE set to ', SQLSTATE,' in OuterProc (exception)' TO
CLIENT;
        RESIGNAL ;
END;
CREATE OR REPLACE PROCEDURE InnerProc()
BEGIN
   DECLARE column_not_found EXCEPTION FOR SQLSTATE '52003';
   MESSAGE 'Hello from InnerProc' TO CLIENT;
   SELECT  'OK';
   SIGNAL column_not_found;
   MESSAGE 'SQLSTATE set to ', SQLSTATE,' in InnerProc (no exception)' TO CLIENT;
   EXCEPTION
      WHEN column_not_found THEN
        MESSAGE 'SQLSTATE set to ', SQLSTATE, ' in InnerProc (exception)' TO
CLIENT;
        --RESIGNAL;
      WHEN OTHERS THEN
        RESIGNAL;
END;
CALL OuterProc();
```

When this example is run using Interactive SQL, the **Results** tab shows the result OK. The **History** tab displays the following:

```
Hello from OuterProc
Hello from InnerProc
SQLSTATE set to 52003 in InnerProc (exception)
```

```
SQLSTATE set to 00000 in OuterProc (no exception)
```

The EXCEPTION clause declares the start of one or more exception handlers. The lines following EXCEPTION do not execute unless an error occurs. Each WHEN clause specifies an exception name (declared with a DECLARE statement) and the statement or statements to be executed in the event of that exception.

The WHEN OTHERS THEN clause specifies the statement(s) to be executed when the exception that occurred does not appear in the preceding WHEN clauses.

In the above example, the statement RESIGNAL passes the exception on to a higher-level exception handler. If WHEN OTHERS THEN is not specified in an exception handler, the default action for any unhandled exception is RESIGNAL.

To pass the `column_not_found` exception to OuterProc, remove the comment indicator from the RESIGNAL statement. This will cause the exception handler in the OuterProc procedure to be invoked.

Additional notes

- The EXCEPTION handler executes, rather than the lines following the SIGNAL statement in InnerProc.
- As the error encountered was an error about a column that cannot be found, the MESSAGE statement included to handle the error executes, and SQLSTATE resets to zero (indicating no errors).
- After the exception handling code executes, control passes back to OuterProc, which proceeds as if no error was encountered.
- Do not use ON EXCEPTION RESUME together with explicit exception handling. The exception handling code is not executed if ON EXCEPTION RESUME is included.
- If the error handling code for the error is a RESIGNAL statement, control returns to the OuterProc procedure with SQLSTATE still set at the value 52003. This is just as if there were no error handling code in InnerProc. Since there is no error handling code in OuterProc, the procedure fails.

11.7.5 Exception handling and nested compound statements

The code following a statement that causes an error executes only if an ON EXCEPTION RESUME clause appears in a procedure definition.

You can use nested compound statements to give you more control over which statements execute following an error and which do not.

The following example illustrates how nested compound statements can be used to control flow.

```
CREATE OR REPLACE PROCEDURE InnerProc()
BEGIN
   BEGIN
      DECLARE column_not_found EXCEPTION FOR SQLSTATE VALUE '52003';
      MESSAGE 'Hello from InnerProc' TO CLIENT;
      SIGNAL column_not_found;
      MESSAGE 'Line following SIGNAL' TO CLIENT;
      EXCEPTION
         WHEN column_not_found THEN
            MESSAGE 'Column not found handling' TO CLIENT;
         WHEN OTHERS THEN
            RESIGNAL;
```

```
   END;
   MESSAGE 'Outer compound statement' TO CLIENT;
END;
CALL InnerProc();
```

The Interactive SQL **History** tab displays the following:

```
Hello from InnerProc
Column not found handling
Outer compound statement
```

When the SIGNAL statement that causes the error is encountered, control passes to the exception handler for the compound statement, and the `Column not found handling` message prints. Control then passes back to the outer compound statement and the `Outer compound statement` message prints.

If an error other than Column not found (SQLSTATE) is encountered in the inner compound statement, the exception handler executes the RESIGNAL statement. The RESIGNAL statement passes control directly back to the calling environment, and the remainder of the outer compound statement is not executed.

Example

This example shows the output of the sa_error_stack_trace system procedure for procedures that use EXCEPTION, RESIGNAL, and nested BEGIN statements:

```
CREATE OR REPLACE PROCEDURE error_reporting_procedure()
BEGIN
    SELECT * FROM sa_error_stack_trace();
END;
CREATE OR REPLACE PROCEDURE proc1()
BEGIN
    BEGIN
        DECLARE v INTEGER = 0;
        SET v = 1 / v;
        EXCEPTION
            WHEN OTHERS THEN
                 CALL proc2();
    END
    EXCEPTION
        WHEN OTHERS THEN
            CALL error_reporting_procedure();
END;
CREATE OR REPLACE PROCEDURE proc2()
BEGIN
    CALL proc3();
END;
CREATE OR REPLACE PROCEDURE proc3()
BEGIN
    RESIGNAL;
END;
CALL proc1();
```

When the `proc1` procedure is called, the following result set is produced:

StackLevel	UserName	ProcName	LineNumber	IsResignal
1	DBA	proc1	8	0
2	DBA	proc2	3	0

StackLevel	UserName	ProcName	LineNumber	IsResignal
3	DBA	proc3	3	1
4	DBA	proc1	5	0

This example shows the output of the sa_error_stack_trace system procedure for procedures that use RESIGNAL and nested BEGIN TRY/CATCH statements:

```
CREATE OR REPLACE PROCEDURE error_reporting_procedure()
BEGIN
    SELECT * FROM sa_error_stack_trace();
END;
CREATE OR REPLACE PROCEDURE proc1()
BEGIN TRY
    BEGIN TRY
        DECLARE v INTEGER = 0;
        SET v = 1 / v;
    END TRY
    BEGIN CATCH
        CALL proc2();
    END CATCH
END TRY
BEGIN CATCH
    CALL error_reporting_procedure();
END CATCH;
CREATE OR REPLACE PROCEDURE proc2()
BEGIN
    CALL proc3();
END;
CREATE OR REPLACE PROCEDURE proc3()
BEGIN
    RESIGNAL;
END;
CALL proc1();
```

When the `proc1` procedure is called, the following result set is produced:

StackLevel	UserName	ProcName	LineNumber	IsResignal
1	DBA	proc1	8	0
2	DBA	proc2	3	0
3	DBA	proc3	3	1
4	DBA	proc1	5	0

11.8 Transactions and savepoints in procedures

SQL statements in a procedure or trigger are part of the current transaction.

SQL statements in a procedure are part of the current transaction. You can call several procedures within one transaction or have several transactions in one procedure.

COMMIT and ROLLBACK are not allowed within any atomic statement.

Triggers are fired due to an INSERT, UPDATE, or DELETE which are atomic statements. COMMIT and ROLLBACK are not allowed in a trigger or in any procedures called by a trigger.

Savepoints can be used within a procedure, but a ROLLBACK TO SAVEPOINT statement can never refer to a savepoint before the atomic operation started. Also, all savepoints within an atomic operation are released when the atomic operation completes.

11.9 Hiding the contents of a procedure, function, trigger, event, or view

Use the SET HIDDEN clause to obscure the contents of a procedure, function, trigger, event, or view.

Prerequisites

You must be the owner of the object, have the ALTER ANY OBJECT system privilege, or have one of the following privileges:

Procedures and functions

ALTER ANY PROCEDURE system privilege

Views

ALTER ANY VIEW system privilege

Events

MANAGE ANY EVENT system privilege

Triggers

- ALTER ANY TRIGGER system privilege
- ALTER privilege on the underlying table and the CREATE ANY OBJECT system privilege
- For triggers on views, you must have the ALTER ANY TRIGGER and ALTER ANY VIEW system privileges

Context

To distribute an application and a database without disclosing the logic contained within procedures, functions, triggers, events, and views, you can obscure the contents of these objects using the SET HIDDEN clause of the ALTER PROCEDURE, ALTER FUNCTION, ALTER TRIGGER, ALTER EVENT and ALTER VIEW statements.

The SET HIDDEN clause obfuscates the contents of the associated objects and makes them unreadable, while still allowing the objects to be used. You can also unload and reload the objects into another database.

The modification is irreversible, and deletes the original text of the object. Preserving the original source for the object outside the database is required.

> **i** Note
>
> Setting the preserve_source_format database option to On causes the database server to save the formatted source from CREATE and ALTER statements on procedures, views, triggers, and events, and put

it in the appropriate system view's source column. In this case both the object definition and the source definition are hidden.

However, setting the preserve_source_format database option to On does *not* prevent the SET HIDDEN clause from deleting the original source definition of the object.

Procedure

Use the appropriate ALTER statement with the SET HIDDEN clause.

Option	Action
Hide an individual object	Execute the appropriate ALTER statement with the SET HIDDEN clause to hide a single procedure, function, trigger, event, or view.
Hide all objects of a specific type	Execute the appropriate ALTER statement with the SET HIDDEN clause in a loop to hide all procedures, functions, triggers, events, or views.

Results

An automatic commit is executed. The object definition is no longer visible. The object can still be directly referenced, and is still eligible for use during query processing.

Example

Execute the following loop to hide all procedures:

```
BEGIN
    FOR hide_lp as hide_cr cursor FOR
        SELECT proc_name, user_name
        FROM SYS.SYSPROCEDURE p, SYS.SYSUSER u
        WHERE p.creator = u.user_id
        AND p.creator NOT IN (0,1,3)
    DO
        MESSAGE 'altering ' || proc_name;
        EXECUTE IMMEDIATE 'ALTER PROCEDURE "' ||
            user_name || '"."' || proc_name
            || '" SET HIDDEN'
    END FOR
END;
```

11.10 Statements allowed in procedures, triggers, events, and batches

Most SQL statements are acceptable in batches, but there are several exceptions.

- ALTER DATABASE (depending on the syntax)
- CONNECT
- CREATE DATABASE
- CREATE DECRYPTED FILE
- CREATE ENCRYPTED FILE
- DISCONNECT
- DROP CONNECTION
- DROP DATABASE
- FORWARD TO
- Interactive SQL statements such as INPUT or OUTPUT
- PREPARE TO COMMIT
- STOP SERVER

You can use COMMIT, ROLLBACK, and SAVEPOINT statements within procedures, triggers, events, and batches with certain restrictions.

In this section:

SELECT statements used in batches [page 285]
You can include one or more SELECT statements in a batch.

EXECUTE IMMEDIATE used in procedures, triggers, user-defined functions, and batches [page 286]
The EXECUTE IMMEDIATE statement allows statements to be constructed using a combination of literal strings (in quotes) and variables.

11.10.1 SELECT statements used in batches

You can include one or more SELECT statements in a batch.

For example:

```
IF EXISTS( SELECT *
           FROM SYSTAB
           WHERE table_name='Employees' )
THEN
   SELECT   Surname AS LastName,
            GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
END IF;
```

The alias for the result set is necessary only in the first SELECT statement, as the server uses the first SELECT statement in the batch to describe the result set.

A RESUME statement is necessary following each query to retrieve the next result set.

11.10.2 EXECUTE IMMEDIATE used in procedures, triggers, user-defined functions, and batches

The EXECUTE IMMEDIATE statement allows statements to be constructed using a combination of literal strings (in quotes) and variables.

For example, the following procedure includes an EXECUTE IMMEDIATE statement that creates a table.

```
CREATE PROCEDURE CreateTableProcedure(
      IN tablename CHAR(128) )
BEGIN
   EXECUTE IMMEDIATE 'CREATE TABLE '
   || tablename
   || '( column1 INT PRIMARY KEY )'
END;
```

The EXECUTE IMMEDIATE statement can be used with queries that return result sets. Use the WITH RESULT SET ON clause with the EXECUTE IMMEDIATE statement to indicate that the statement returns a result set. The default behavior is that the statement does not return a result set. Specifying WITH RESULT SET ON or WITH RESULT SET OFF affects both what happens when the procedure is created, as well as what happens when the procedure is executed.

Consider the following procedure:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET OFF 'SELECT 1';
END;
```

While the procedure definition does not include a RESULT SET clause, the database server tries to determine if the procedure generates one. Here, the EXECUTE IMMEDIATE statement specifies that a result set is not generated. Consequently, the database server defines the procedure with no result set columns, and no rows exist in the SYSPROCPARM system view for this procedure. A DESCRIBE on a CALL to this procedure would return no result columns. If an Embedded SQL application used that information to decide whether to open a cursor or execute the statement, it would execute the statement and then return an error.

As a second example, consider a modified version of the above procedure:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET ON 'SELECT 1';
END;
```

Here, the WITH RESULT SET ON clause causes a row to exist for this procedure in the SYSPROCPARM system view. The database server does not know what the result set looks like because the procedure is using EXECUTE IMMEDIATE, but it knows that one is expected, so the database server defines a dummy result set column in SYSPROCPARM to indicate this, with a name of "expression" and a type of SMALLINT. Only *one* dummy result set column is created; the server cannot determine the number and type of each result set column when an EXECUTE IMMEDIATE statement is being used. Consequently, consider this slightly modified example:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
```

```
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET ON 'SELECT 1, 2, 3';
END;
```

Here, while the SELECT returns a result set of three columns, the server still only places one row in the SYSPROCPARM system view. Hence, this query

```
SELECT * FROM test_result_clause();
```

fails with SQLCODE -866, as the result set characteristics at run time do not match the placeholder result in SYSPROCPARM.

To execute the query above, you can explicitly specify the names and types of the result set columns as follows:

```
SELECT * FROM test_result_clause() WITH (x INTEGER, y INTEGER, z INTEGER);
```

At execution time, if WITH RESULT SET ON is specified, the database server handles an EXECUTE IMMEDIATE statement that returns a result set. However, if WITH RESULT SET OFF is specified or the clause is omitted, the database server *still* looks at the type of the first statement in the parsed string argument. If that statement is a SELECT statement, it returns a result set. Hence, in the second example above:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET OFF 'SELECT 1';
END;
```

this procedure can be called successfully from Interactive SQL. However, if you change the procedure so that it contains a batch, rather than a single SELECT statement:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET OFF
    'begin declare v int; set v=1; select v; end';
END;
```

then a CALL of the test_result_clause procedure returns an error (SQLCODE -946, SQLSTATE 09W03).

This last example illustrates how you can construct a SELECT statement as an argument of an EXECUTE IMMEDIATE statement within a procedure, and have that procedure return a result set.

```
CREATE PROCEDURE DynamicResult(
   IN Columns LONG VARCHAR,
   IN TableName CHAR(128),
   IN Restriction LONG VARCHAR DEFAULT NULL )
BEGIN
    DECLARE Command LONG VARCHAR;
    SET Command = 'SELECT ' || Columns || ' FROM ' || TableName;
    IF ISNULL( Restriction,'') <> '' THEN
         SET Command = Command || ' WHERE ' || Restriction;
    END IF;
    EXECUTE IMMEDIATE WITH RESULT SET ON Command;
END;
```

If the procedure above is called as follows:

```
CALL DynamicResult(
   'table_id,table_name',
   'SYSTAB',
   'table_id <= 10');
```

it yields the following result:

table_id	table_name
1	ISYSTAB
2	ISYSTABCOL
3	ISYSIDX
...	...

The CALL above correctly returns a result set, even though the procedure uses EXECUTE IMMEDIATE. Some server APIs, such as ODBC, use a PREPARE-DESCRIBE-EXECUTE-OR-OPEN combined request that either executes or opens the statement, depending on if it returns a result set. Should the statement be opened, the API or application can subsequently issue a DESCRIBE CURSOR to determine what the actual result set looks like, rather than rely on the content of the SYSPROCPARM system view from when the procedure was created. Both DBISQL and DBISQLC use this technique. In these cases, a CALL of the procedure above executes without an error. However, application interfaces that rely on the statement's DESCRIBE results will be unable to handle an arbitrary statement.

In ATOMIC compound statements, you cannot use an EXECUTE IMMEDIATE statement that causes a COMMIT, as COMMITs are not allowed in that context.

12 Automate Tasks Using Schedules and Events

Use scheduling and event handling features to automate database administration and other tasks.

In this section:

Task automation using schedules and events [page 289]
Automate database administration tasks with schedules and events

12.1 Task automation using schedules and events

Automate database administration tasks with schedules and events

Automate tasks

Automate routine tasks by adding an **event** to a database, and providing a schedule for the event. Whenever one of the times in the schedule passes, the database server runs a sequence of actions called an **event handler**.

To schedule back ups, use a maintenance plan..

Trigger tasks

Database administration also requires taking action when certain conditions occur. For example, it may be appropriate to email a notification to a system administrator when a disk containing the transaction log is filling up so that the administrator can handle the situation. These tasks too can be automated by defining event handlers for one of a set of **system events**.

In this section:

Events [page 290]
You can automate routine tasks by adding an event to a database, and providing a schedule for the event.

Schedules [page 290]
By scheduling activities you can ensure that a set of actions is executed at a set of preset times.

System events [page 291]
Each system event provides a hook on which you can program a set of actions.

Event handlers [page 294]
: Event handlers execute on a separate connection from the action that triggered the event, and do not interact with client applications.

12.1.1 Events

You can automate routine tasks by adding an event to a database, and providing a schedule for the event.

The following types of events are supported:

Scheduled events

Scheduled events have an associated schedule and execute at specified times.

System events

System events are associated with a particular type of condition that is tracked by the database server.

Manual events

Manual events are fired explicitly using the TRIGGER EVENT statement.

User trace events

User trace events are used to log information about an application to an event tracing session. These events are visible to all connections to a database.

After each execution of an event handler, a COMMIT occurs if no errors occurred. A ROLLBACK occurs if there was an error.

12.1.2 Schedules

By scheduling activities you can ensure that a set of actions is executed at a set of preset times.

The scheduling information and the event handler are both stored in the database itself.

Although this is not usually necessary, you can define complex schedules by associating more than one schedule with a named event. For example, a retail outlet might want an event to occur once per hour during hours of operation, where the hours of operation vary based on the day of the week. You can achieve the same effect by defining multiple events, each with its own schedule, and by calling a common stored procedure.

When scheduling events, you can use either full-length English day names (Monday, Tuesday, and so on) or the abbreviated forms of the day (Mon, Tue, and so on). You must use the full-length English day names if you want the day names to be recognized by a server running in a language other than English.

The following examples give some ideas for scheduled actions that may be useful.

Example

Summarize orders at the end of each business day.

```
/*  This example uses iqdemo.db. */
CREATE TABLE OrderSummary(c1 date, c2 int);
CREATE EVENT SummarizeSchedule
START TIME '6:00 pm'on ('Mon', 'Tue', 'Wed', 'Thu', 'Fri')
HANDLER
  BEGIN
    INSERT INTO DBA.OrderSummary
      SELECT MAX(OrderDate),
      COUNT(*)
  FROM GROUPO.SalesOrders
  WHERE OrderDate = current date
END;
```

In this section:

Schedule definition [page 291]
To permit flexibility, schedule definitions have several components to them.

12.1.2.1 Schedule definition

To permit flexibility, schedule definitions have several components to them.

Name

Each schedule definition has a name. You can assign more than one schedule to a particular event, which can be useful in designing complex schedules.

Start time

You can define a start time for the event, which is the time when execution begins.

Range

As an alternative to a start time, you can specify a range of times for which the event is active. The event occurs between the start and end time specified. Frequency is determined by the specified recurrence.

Recurrence

Each schedule can recur. The event is triggered on a frequency that can be given in hours, minutes, or seconds on a set of days that can be specified as days of the week or days of the month. Recurring events include an EVERY or ON clause.

12.1.3 System events

Each system event provides a hook on which you can program a set of actions.

The database server tracks the events for you, and executes the actions (as defined in the event handler) when the system event satisfies a provided **trigger condition**.

By defining event handlers to execute when a chosen system event type occurs and satisfies a trigger condition that you define, you can improve the security and safety of your data, and help ease administration. The actions of an event handler are committed if no error is detected during execution, and rolled back if errors are detected.

In this section:

System event types [page 292]
There are several system event types.

Trigger conditions for events [page 293]
Each event definition has a system event associated with it, and one or more trigger conditions.

12.1.3.1 System event types

There are several system event types.

Audit events

BackupEnd

You can use the BackupEnd event type to take action at the end of a backup.

Connection events

When a connection is made (Connect) or when a connection attempt fails (ConnectFailed). You may want to use these events for security purposes. As an alternative to a connect event handler, you may want to consider using a login procedure.

DatabaseStart

You can use the DatabaseStart event type to take action when a database is started.

Deadlock

You can use the Deadlock event to take action when a deadlock occurs. The event handler can use the sa_report_deadlocks procedure to obtain information about the conditions that led to the deadlock. When using the Deadlock event, you should configure the database server to capture deadlock information by setting the log_deadlocks option to On, and by enabling the RememberLastStatement feature using sa_server_option or the -zl server option.

Deadlock events fire for connection deadlocks and thread deadlocks. A deadlock event provides no information beyond what is available via the sa_report_deadlocks system procedure. However, using this event allows you to act on the deadlock in a timely manner. A quick response may be important since the amount of deadlock-related information the database server maintains is limited.

Disconnect

You can use the Disconnect event to take action when a user or application disconnects.

Free disk space

Tracks the available disk space on the device holding the database file (DBDiskSpace), the transaction log file (LogDiskSpace), or temporary file (TempDiskSpace).

You may want to use disk space events to alert administrators of a disk space shortage.

You can specify the -fc option when starting the database server to implement a callback function when the database server encounters a file system full condition.

File size

The file reaches a specified size. This can be used for the database file (GrowDB), the transaction log (GrowLog), or the temporary file (GrowTemp).

You may want to use file size events to track unusual actions on the database, or monitor bulk operations.

GlobalAutoincrement

When the number of remaining values for a column defined with GLOBAL AUTOINCREMENT is less than one percent of its range, the GlobalAutoincrement event fires. This can be used to request a new value for the global_database_id option based on the table and number of remaining values that are supplied as parameters to this event. To get the remaining values for the table within the event, use the EVENT_PARAMETER function with the RemainingValues and TableName parameters. RemainingValues returns the number of remaining values that can be generated for the column, while TableName returns the table containing the GLOBAL AUTOINCREMENT column that is near the end of its range.

RAISERROR error

When a RAISERROR statement is executed, you can use the RAISERROR event type to take actions. The error number used in the RAISERROR statement can be determined within the event handler using the EVENT_CONDITION function (for example, `EVENT_CONDITION( 'ErrorNumber' )`).

Idle time

The database server has been idle for a specified time (ServerIdle). You may want to use this event type to perform routine maintenance operations at quiet times.

12.1.3.2 Trigger conditions for events

Each event definition has a system event associated with it, and one or more trigger conditions.

The event handler is triggered when the trigger conditions for the system event are satisfied.

The trigger conditions are included in the WHERE clause of the CREATE EVENT statement, and can be combined using the AND keyword. Each trigger condition is of the following form:

event_condition `( <condition-name> ) <comparison-operator> <value>`

The `<condition-name>` argument is one of a set of preset strings, which are appropriate for different event types. For example, you can use DBSize (the database file size in megabytes) to build a trigger condition suitable for the GrowDB system event. The database server does not check that the condition-name matches the event type: it is your responsibility to ensure that the condition is meaningful in the context of the event type.

Example

- Notify an administrator of a possible attempt to break into the database. This example uses iqdemo.db:

```
create event SecurityCheck
type ConnectFailed
handler
begin
```

```
declare num_failures int;
declare mins int;
insert into FailedConnections( log_time )
values ( current timestamp );
select count( * ) into num_failures
from FailedConnections
where log_time >= dateadd(minute, -5, current timestamp );
if( num_failures >= 3 ) then
  select datediff( minute, last_notification, current timestamp ) into mins
from Notification;
  if( mins > 30 ) then
    update Notification
    set last_notification = current timestamp;
end if
end if
end;
```

12.1.4 Event handlers

Event handlers execute on a separate connection from the action that triggered the event, and do not interact with client applications.

They execute with the privileges of the creator of the event.

Event handlers, whether for scheduled events or for system event handling, contain compound statements, and are similar in many ways to stored procedures. You can add loops, conditional execution, and so on.

After each execution of an event handler, a COMMIT occurs if no errors occurred. A ROLLBACK occurs if there was an error.

Context information for event handlers

Unlike stored procedures, event handlers do not take any arguments. You can use the EVENT_PARAMETER function to access information about the context in which an event was triggered. The information returned includes the connection ID and user ID that caused an event to be triggered, and the event name and the number of times it has been executed.

Test event handlers

During development, you want event handlers to be triggered at convenient times. You can use the TRIGGER EVENT statement to explicitly cause an event to execute, even when the trigger condition or scheduled time has not occurred. However, TRIGGER EVENT does not cause disabled event handlers to be executed.

While it is not good practice to develop event handlers on a production database, you can disable event handlers explicitly using the ALTER EVENT statement.

Code sharing

It can be useful to use a single set of actions to handle multiple events. For example, you may want to take a notification action if disk space is limited on any of the devices holding the database or log files. To do this, create a stored procedure and call it in the body of each event handler, passing any needed context information as parameters to the procedure.

Debug event handlers

Debugging event handlers is very similar to debugging stored procedures. The event handlers appear in the events list.

Hide event handlers

You can use the ALTER EVENT statement with the SET HIDDEN clause to hide the definition of an event handler. Specifying the SET HIDDEN clause results in the permanent obfuscation of the event handler definition stored in the action column of the ISYSEVENT system table.

Limit active events

You can also determine how many instances of a particular event handler are currently active using EVENT_PARAMETER function with the NumActive context name. This function is useful to limit an event handler so that only one instance executes at any given time.

In this section:

How the database server checks for system events [page 296]
System events are classified according to their **event type**, as specified in the CREATE EVENT statement.

How the database server checks for scheduled events [page 296]
The calculation of scheduled event times is done when the database server starts, and each time a scheduled event handler completes.

How event handlers are executed [page 297]
When an event handler is triggered, a temporary internal connection is made on which the event handler is executed.

Hiding an event handler [page 297]
Hide the definition for an event handler using the ALTER EVENT statement.

12.1.4.1 How the database server checks for system events

System events are classified according to their **event type**, as specified in the CREATE EVENT statement.

There are two kinds of event types:

Active event types

Some event types are the result of action by the database server itself. These active event types include growing database files, or the start and end of different database actions (BackupEnd and so on) or RAISERROR.

When the database server takes the action, it checks to see whether the trigger conditions defined in the WHERE clause are satisfied, and if so, triggers any events defined for that event type.

Polled event types

Some event types, such as free disk space types (DBDiskSpace and so on) and IdleTime type, are not triggered solely by database actions.

For these types of events, the database server polls every thirty seconds, starting approximately thirty seconds after the database server is started.

For the IdleTime event type, the database server checks whether the server has been idle for the entire thirty seconds. If no requests have started and none are currently active, it adds the idle check interval time in seconds to the idle time total; otherwise, the idle time total is reset to 0. The value for IdleTime is therefore always a multiple of thirty seconds. When IdleTime is greater than the interval specified in the trigger condition, event handlers associated with IdleTime are fired.

12.1.4.2 How the database server checks for scheduled events

The calculation of scheduled event times is done when the database server starts, and each time a scheduled event handler completes.

The calculation of the next scheduled time is based on the increment specified in the schedule definition, with the increment being added to the previous start time. If the event handler takes longer to execute than the specified increment, so that the next time is earlier than the current time, the database server increments until the next scheduled time is in the future.

For example, an event handler that takes sixty-five minutes to execute and is requested to run every hour between 9:00 and 5:00 will run every two hours, at 9:00, 11:00, 1:00, and so on.

To run a process such that it operates between 9:00 and 5:00 and delays for some period before the next execution, you could define a handler to loop until its completion time has passed, with a WAITFOR statement between each iteration.

If you are running a database server intermittently, and it is not running at a scheduled time, the event handler does not run at startup. Instead, the next scheduled time is computed at startup. If, for example, you schedule a backup to take place every night at one o'clock, but regularly shut down the database server at the end of each work day, the backup never takes place.

If the next scheduled execution of an event is more than one hour away, the database server will recalculate its next scheduled time on an hourly basis. This allows events to fire when expected when the system clock is adjusted because of a change to or from Daylight Savings Time.

12.1.4.3 How event handlers are executed

When an event handler is triggered, a temporary internal connection is made on which the event handler is executed.

The handler is not executed on the connection that caused the handler to be triggered, so statements such as MESSAGE...TO CLIENT, which interact with the client application, are not meaningful within event handlers. Similarly, statements that return result sets are not permitted.

The temporary connection on which the handler is executed does not count towards the connection limit for licensing purposes, and the procedure specified by the login_procedure option is not executed for event connections.

Event handlers execute on a separate connection, with the privileges of the event owner. You can also call a procedure from within the event handler, in which case the procedure executes with the privileges of procedure owner. The separate connection for the event handler does not count towards the ten-connection limit of the personal database server.

Any event errors are logged to the database server message log.

> **i** Note
>
> The transaction in an event handler is committed if no errors are detected during execution, and rolled back if errors are detected.

12.1.4.4 Hiding an event handler

Hide the definition for an event handler using the ALTER EVENT statement.

Prerequisites

You must have either the MANAGE ANY EVENT or ALTER ANY OBJECT system privilege.

Procedure

1. Connect to the database.
2. Execute an ALTER EVENT statement with the SET HIDDEN clause:

```
ALTER EVENT <event-name> SET HIDDEN
```

Results

The event handler is permanently obfuscated in the event handler definition that is stored in the action column of the ISYSEVENT system table.

Example

The following statement hides the definition of the event handler secret_event:

```
ALTER EVENT secret_event SET HIDDEN
```

In this section:

Retrieving Information About an Event or Schedule [page 298]
SAP IQ stores information about events, system events, and schedules in the system tables SYSEVENT, SYSEVENTTYPE, and SYSSCHEDULE.

12.1.4.4.1 Retrieving Information About an Event or Schedule

SAP IQ stores information about events, system events, and schedules in the system tables SYSEVENT, SYSEVENTTYPE, and SYSSCHEDULE.

When you alter an event using the ALTER EVENT statement, specify the event name and, optionally, the schedule name. When you trigger an event using the TRIGGER EVENT statement, specify the event name.

You can list event names by querying the SYSEVENT system table:

```
SELECT event_id, event_name FROM SYSEVENT
```

You can list schedule names by querying the SYSSCHEDULE system table:

```
SELECT event_id, sched_name FROM SYSSCHEDULE
```

Each event has a unique event ID. Use the event_id columns of SYSEVENT and SYSSCHEDULE to match the event to the associated schedule.

13 Audit Database Events

Auditing records database events in the transaction log.

In this section:

Database activity audits [page 299]
Auditing tracks all of the activity performed on a database.

13.1 Database activity audits

Auditing tracks all of the activity performed on a database.

Each database has an associated transaction log file that is used for database recovery and contains a record of transactions executed against a database.

The transaction log stores all executed data definition statements, along with the user ID that executed them. It also stores all updates, deletes, and inserts and which user executed those statements. However, this information is insufficient for some auditing purposes. By default, the transaction log does not contain the time of the event. It only contains the order in which events occurred. It does not contain failed events or select statements.

When you use auditing, additional data is saved in the audit log, which is in the transaction log, the user-defined ETD log, or the system event log. This data potentially includes:

- All login attempts (successful and failed), including the computer name.
- Accurate timestamps of all events (to a resolution of milliseconds).
- All permissions checks (successful and failed), including the object on which the permission was checked (if applicable).
- All actions that require system privileges.
- All executions of xp_cmdshell system procedure.

You cannot stop using a transaction log while auditing is enabled for a database and the audit log is the transaction log. In this case, to turn off the transaction log, either first turn off auditing or change the audit_log option to specify FILE or SYSLOG.

Auditing individual connections

Once you have enabled auditing for a database, you can set the temporary conn_auditing database option in the database login procedure to enable connection-specific auditing. You can enable auditing based on information such as the IP address of the client computer or the type of connection.

If you do not set the conn_auditing option in the login procedure, then the option is on by default.

The following example shows an excerpt from a login procedure that enables auditing for all connections to the database, except those made by a user named DBA:

```
DECLARE usr VARCHAR(128)
SELECT CONNECTION_PROPERTY( 'Userid' ) INTO usr;
IF usr != 'DBA' THEN
   SET TEMPORARY OPTION conn_auditing='On'
ELSE
   SET TEMPORARY OPTION conn_auditing='Off'
END IF;
```

Audit comments

Add comments to the audit trail by using the sa_audit_string system stored procedure. It takes a single argument, which is a string of up to 200 bytes. For example:

```
CALL sa_audit_string( 'Started audit testing here.' );
```

This comment is stored in the audit log as an audit statement.

In this section:

Configuring auditing (SQL) [page 300]
Configure auditing for security-related information in the transaction log. Auditing is off by default.

Retrieving auditing information (dbtran utility) [page 301]
Retrieve auditing information from a transaction log.

13.1.1 Configuring auditing (SQL)

Configure auditing for security-related information in the transaction log. Auditing is off by default.

Prerequisites

You must have the SET ANY SECURITY OPTION system privilege.

Procedure

1. Connect to your database.
2. Execute the following statement to turn on auditing:

   ```
   SET OPTION PUBLIC.auditing = 'On';
   ```

3. Execute the following statement to force logging to a transaction log:

   ```
   SET OPTION PUBLIC.audit_log = 'TRANSLOG';
   ```

4. To specify which types of auditing information you want to enable, run the following system procedure:

   ```
   CALL sa_enable_auditing_type( 'all' );
   ```

   You can control the type of auditing information that is collected by replacing *all* with the types of auditing you want to enable.

5. Execute the following statement to turn off auditing:

   ```
   SET OPTION PUBLIC.auditing = 'Off';
   ```

   To specify which types of auditing information you want to disable, use the following system procedure:

   ```
   CALL sa_disable_auditing_type( 'all' );
   ```

   You can stop collecting specific types of auditing information by replacing *all* with the types of auditing you want to disable.

Results

The auditing behavior is configured.

13.1.2 Retrieving auditing information (dbtran utility)

Retrieve auditing information from a transaction log.

Prerequisites

You must have the BACKUP DATABASE system privilege to retrieve auditing information from a running database server. No privileges are required to retrieve auditing information from a database transaction log file.

The database must have the auditing database option turned on.

Context

The dbtran utility translates a transaction log into a SQL script file. When the dbtran utility is run with the -g option, it adds auditing information (as comments) and orders the SQL statements chronologically. Do not apply this output to a database, because running statements in chronological order does not guarantee that the resulting database will be identical.

Procedure

1. Connect to the database.
2. Run the dbtran utility with the -g option against a running database server or against a database transaction log file.

Option	Action
Retrieve auditing information from a running database server	Run the following command: `dbtran -g -c <connection-string> -n <SQL-file>`
Retrieve auditing information from a transaction log file	Shut down the database server to ensure the transaction log file is available. Run the following command: `dbtran -g <transaction-log> <SQL-file>`

Results

Auditing information is retrieved and saved in a SQL script file.

14 Access Remote Data

SAP IQ can access data located on separate servers, both SAP and non-SAP, as if the data were stored on the local server.

You can use this feature to migrate data into an SAP IQ database or to query data across databases.

For information on remote transactions, see Remote Transactions [page 243].

In this section:

SAP IQ and Remote Data [page 303]
SAP IQ remote data access gives you access to data in other data sources. You can use this feature to migrate data into an SAP SQL Anywhere database or query data across databases.

Internal Operations [page 346]
This section describes the underlying steps that SAP IQ performs on remote servers on behalf of client applications.

Remote Data Access Troubleshooting [page 349]
This section provides some suggestions for troubleshooting access to remote servers.

14.1 SAP IQ and Remote Data

SAP IQ remote data access gives you access to data in other data sources. You can use this feature to migrate data into an SAP SQL Anywhere database or query data across databases.

In this section:

Characteristics of Open Client and jConnect connections [page 304]
When SAP IQ is serving applications over TDS, it automatically sets relevant database options to values compatible with SAP Adaptive Server Enterprise default behavior. These options are set temporarily, for the duration of the connection only. The client application can override them at any time.

Requirements for Accessing Remote Data [page 306]
There are several basic elements required to access remote data.

Remote Servers [page 327]
Before remote objects can be mapped to a local proxy table, define the remote server where the remote object is located.

External Logins [page 338]
SAP IQ uses the names and passwords of its clients when it connects to a remote server on behalf of those clients. However, this behavior can be overridden by creating external logins.

Proxy tables [page 339]
Use a proxy table to access any object (including tables, views, and materialized views) that the remote database exports as a candidate for a proxy table.

Joins between remote tables [page 343]
You can use joins between proxy tables and remote tables.

Joins between tables from multiple local databases [page 344]
A database server may have several local databases running at one time. By defining tables in other local SAP IQ databases as remote tables, you can perform cross-database joins.

Native statements and remote servers [page 345]
Use the FORWARD TO statement to send one or more statements to the remote server in its native syntax.

Remote Procedure Calls (RPCs) [page 345]
SAP IQ users can issue procedure calls to remote servers that support the feature.

14.1.1 Characteristics of Open Client and jConnect connections

When SAP IQ is serving applications over TDS, it automatically sets relevant database options to values compatible with SAP Adaptive Server Enterprise default behavior. These options are set temporarily, for the duration of the connection only. The client application can override them at any time.

Default Settings

The database options set on connection using TDS include:

Option	Set to
allow_nulls_by_default	Off
ansinull	Off
chained	Off
close_on_endtrans	Off
date_format	YYYY-MM-DD
date_order	MDY
escape_character	Off
isolation_level	1
on_tsql_error	Continue
quoted_identifier	Off
time_format	HH:NN:SS.SSS
timestamp_format	YYYY-MM-DD HH:NN:SS.SSS
tsql_variables	On

How the Startup Options are Set

The default database options are set for TDS connections using a system procedure named `sp_tsql_environment`. This procedure sets the following options:

```
SET TEMPORARY OPTION allow_nulls_by_default='Off';
SET TEMPORARY OPTION ansinull='Off';
SET TEMPORARY OPTION chained='Off';
SET TEMPORARY OPTION close_on_endtrans='Off';
SET TEMPORARY OPTION date_format='YYYY-MM-DD';
SET TEMPORARY OPTION date_order='MDY';
SET TEMPORARY OPTION escape_character='Off';
SET TEMPORARY OPTION isolation_level='1';
SET TEMPORARY OPTION on_tsql_error='Continue';
SET TEMPORARY OPTION quoted_identifier='Off';
SET TEMPORARY OPTION time_format='HH:NN:SS.SSS';
SET TEMPORARY OPTION timestamp_format='YYYY-MM-DD HH:NN:SS.SSS';
SET TEMPORARY OPTION tsql_variables='On';
```

> ⚠ Caution
>
> Do not alter the `sp_tsql_environment` procedure. It is for system use only.

The procedure sets options only for connections that use the TDS communications protocol. This includes Open Client and JDBC connections using jConnect. Other connections (ODBC and embedded SQL) have the default settings for the database.

Although SAP IQ allows longer user names and passwords, TDS client names and passwords cannot exceed 30 bytes. If your password or user ID is longer than 30 bytes, attempts to connect over TDS (for example, using jConnect) return an invalid user or password error.

In this section:

Changing the option settings for TDS connections [page 305]
When SAP IQ is serving applications over TDS, it automatically sets relevant database options to values compatible with SAP Adaptive Server Enterprise default behavior. You can change the options for TDS connections at any time.

14.1.1.1 Changing the option settings for TDS connections

When SAP IQ is serving applications over TDS, it automatically sets relevant database options to values compatible with SAP Adaptive Server Enterprise default behavior. You can change the options for TDS connections at any time.

Procedure

1. Create a procedure that sets the database options you want.
2. Set the `login_procedure` option to the name of the new procedure.

Results

Future connections use the procedure. You can configure the procedure differently for different user IDs.

14.1.2 Requirements for Accessing Remote Data

There are several basic elements required to access remote data.

In this section:

Server classes for remote data access [page 306]
The server class you specify in the CREATE SERVER statement determines the behavior of a remote connection.

14.1.2.1 Server classes for remote data access

The server class you specify in the CREATE SERVER statement determines the behavior of a remote connection.

The server classes give SAP IQ detailed server capability information. SAP IQ formats SQL statements specific to a server's capabilities.

All server classes are ODBC-based. Each server class has a set of unique characteristics that you need to know to configure the server for remote data access. You should refer to information generic to the server class category and also to the information specific to the individual server class.

The server classes include:

- ADSODBC
- ASEODBC
- DB2ODBC
- HANAODBC
- IQODBC
- MSACCESSODBC
- MSSODBC
- MYSQLODBC
- ODBC
- ORAODBC
- SAODBC
- ULODBC

> **i Note**
>
> When using remote data access, if you use an ODBC driver that does not support Unicode, then character set conversion is not performed on data coming from that ODBC driver.

In this section:

ODBC external server definitions [page 307]
The most common way of defining an ODBC-based remote server is to base it on an ODBC data source. To do this, you can create a data source using the ODBC Data Source Administrator.

USING clause in the CREATE SERVER statement [page 308]
You must issue a separate CREATE SERVER statement for each remote SAP IQ database you intend to access.

Server class SAODBC [page 309]
A remote server with server class SAODBC is a SAP IQ database server.

Server class ADSODBC [page 309]
When you execute a CREATE TABLE statement, SAP IQ automatically converts the data types to the corresponding SAP Advantage Database Server data types. The following table describes the SAP IQ to SAP Advantage Database Server data type conversions.

Server class ASEODBC [page 310]
A remote server with server class ASEODBC is an Adaptive Server Enterprise (version 10 and later) database server.

Server class DB2ODBC [page 313]
A remote server with server class DB2ODBC is an IBM DB2 database server.

Server class HANAODBC [page 315]
A remote server with server class HANAODBC is an SAP HANA database server.

Server class IQODBC [page 316]
A remote server with server class IQODBC is an SAP IQ database server.

Server class MSACCESSODBC [page 316]
Microsoft Access databases are stored in a `.mdb` file. Using the ODBC manager, create an ODBC data source and map it to one of these files.

Server class MSSODBC [page 318]
The server class MSSODBC is used to access Microsoft SQL Server through one of its ODBC drivers.

Server class MYSQLODBC [page 320]
When you execute a CREATE TABLE statement, SAP IQ automatically converts the data types to the corresponding Oracle MySQL data types.

Server class ODBC [page 321]
ODBC data sources that do not have their own server class use ODBC server class.

Server class ORAODBC [page 325]
A remote server with server class ORAODBC is an Oracle Database version 8.0 or later.

14.1.2.1.1 ODBC external server definitions

The most common way of defining an ODBC-based remote server is to base it on an ODBC data source. To do this, you can create a data source using the ODBC Data Source Administrator.

When using remote data access, if you use an ODBC driver that does not support Unicode, then character set conversion is not performed on data coming from that ODBC drive

Once you have defined the data source, the USING clause in the CREATE SERVER statement should refer to the ODBC Data Source Name (DSN).

For example, to configure an IBM DB2 server named mydb2 whose data source name is also mydb2, use:

```
CREATE SERVER mydb2
CLASS 'DB2ODBC'
USING 'mydb2';
```

The driver used must match the bitness of the database server.

On Windows, you must also define a System Data Source Name (System DSN) with a bitness matching the database server. For example, use the 32-bit ODBC Data Source Administrator to create a 32-bit System DSN. A User DSN does not have bitness.

Using connection strings instead of data sources

An alternative, which avoids using data source names, is to supply a connection string in the USING clause of the CREATE SERVER statement. To do this, you must know the connection parameters for the ODBC driver you are using. For example, a connection to a SAP IQ database server may be as follows:

```
CREATE SERVER TestSA
CLASS 'SAODBC'
USING 'DRIVER=SQL
                          Anywhere 17;HOST=myhost;Server=TestSA;DBN=sample';
```

This defines a connection to a database server named TestSA, running on a computer called myhost, and a database named sample using the TCP/IP protocol.

14.1.2.1.2 USING clause in the CREATE SERVER statement

You must issue a separate CREATE SERVER statement for each remote SAP IQ database you intend to access.

For example, if a SAP IQ server named TestSA is running on the computer Banana and owns three databases (db1, db2, db3), you would set up the remote servers similar to this:

```
CREATE SERVER TestSAdb1
CLASS 'SAODBC'
USING 'DRIVER=SQL
                          Anywhere 17;HOST=Banana;Server=TestSA;DBN=db1';
CREATE SERVER TestSAdb2
CLASS 'SAODBC'
USING 'DRIVER=SQL
                          Anywhere 17;HOST=Banana;Server=TestSA;DBN=db2';
CREATE SERVER TestSAdb3
CLASS 'SAODBC'
USING 'DRIVER=SQL
                          Anywhere 17;HOST=Banana;Server=TestSA;DBN=db3';
```

If you do not specify a database name, the remote connection uses the remote SAP IQ server default database.

14.1.2.1.3 Server class SAODBC

A remote server with server class SAODBC is a SAP IQ database server.

No special requirements exist for the configuration of a SAP IQ data source.

To access SAP IQ database servers that support multiple databases, create an ODBC data source name defining a connection to each database. Execute a CREATE SERVER statement for each of these ODBC data source names.

Example

Supply a connection string in the USING clause of the CREATE SERVER statement to connect to a SAP IQ database.

```
CREATE SERVER TestSA
CLASS 'SAODBC'
USING 'DRIVER=SQL
                            Anywhere 17;HOST=myhost;Server=TestSA;DBN=sample';
```

14.1.2.1.4 Server class ADSODBC

When you execute a CREATE TABLE statement, SAP IQ automatically converts the data types to the corresponding SAP Advantage Database Server data types. The following table describes the SAP IQ to SAP Advantage Database Server data type conversions.

SAP IQ data type	Advantage Database Server default data type
BIT	Logical
VARBIT(<n>)	Binary(<n>)
LONG VARBIT	Binary(2G)
TINYINT	Integer
SMALLINT	Integer
INTEGER	Integer
BIGINT	Numeric(32)
UNSIGNED TINYINT	Numeric(11)
UNSIGNED SMALLINT	Numeric(11)
UNSIGNED INTEGER	Numeric(11)
UNSIGNED BIGINT	Numeric(32)
CHAR(<n>)	Character(<n>)
VARCHAR(<n>)	VarChar(<n>)

SAP IQ data type	Advantage Database Server default data type
LONG VARCHAR	VarChar(65000)
NCHAR(<n>)	NChar(<n>)
NVARCHAR(<n>)	NVarChar(<n>)
LONG NVARCHAR	NVarChar(32500)
BINARY(<n>)	Binary(<n>)
VARBINARY(<n>)	Binary(<n>)
LONG BINARY	Binary(2G)
DECIMAL(<precision>, <scale>)	Numeric(<precision>+3)
NUMERIC(<precision>, <scale>)	Numeric(<precision>+3)
SMALLMONEY	Money
MONEY	Money
REAL	Double
DOUBLE	Double
FLOAT(<n>)	Double
DATE	Date
TIME	Time
TIMESTAMP	TimeStamp
TIMESTAMP WITH TIME ZONE	Char(254)
BIGDATE	datetime
BIGDATETIME	datetime
XML	Binary(2G)
ST_GEOMETRY	Binary(2G)
UNIQUEIDENTIFIER	Binary(2G)

14.1.2.1.5 Server class ASEODBC

A remote server with server class ASEODBC is an Adaptive Server Enterprise (version 10 and later) database server.

SAP IQ requires the installation of the Adaptive Server Enterprise ODBC driver and Open Client connectivity libraries to connect to a remote Adaptive Server Enterprise database server with class ASEODBC.

Notes

- Open Client should be version 11.1.1, EBF 7886 or later. Install Open Client and verify connectivity to the Adaptive Server Enterprise server before you install ODBC and configure SAP IQ.
  The most recent version of the SAP Adaptive Server Enterprise ODBC driver that has been tested is SDK 15.7 SP110.
- The local setting of the quoted_identifier option controls the use of quoted identifiers for Adaptive Server Enterprise. For example, if you set the quoted_identifier option to Off locally, then quoted identifiers are turned off for Adaptive Server Enterprise.
- Configure a user data source in the **Configuration Manager** with the following attributes:

  General tab

  Type any value for **Data Source Name**. This value is used in the USING clause of the CREATE SERVER statement.

  The server name should match the name of the server in the interfaces file.

  Advanced tab

  Click the **Application Using Threads** and **Enable Quoted Identifiers** options.

  Connection tab

  Set the charset field to match your SAP IQ character set.

  Set the language field to your preferred language for error messages.

  Performance tab

  Set the **Prepare Method** to **2-Full**.

  Set the **Fetch Array Size** as large as possible for the best performance. This increases memory requirements since this is the number of rows that must be cached in memory. Adaptive Server Enterprise recommends using a value of 100.

  Set **Select Method** to **0-Cursor**.

  Set **Packet Size** to as large a value as possible. Adaptive Server Enterprise recommends using a value of -1.

  Set **Connection Cache** to 1.

Data type conversions: ODBC and Adaptive Server Enterprise

When you execute a CREATE TABLE statement, SAP IQ automatically converts the data types to the corresponding Adaptive Server Enterprise data types. The following table describes the SAP IQ to Adaptive Server Enterprise data type conversions.

SAP IQ data type	Adaptive Server Enterprise default data type
BIT	bit
VARBIT(<n>)	if (n <= 255) varbinary(<n>) else image
LONG VARBIT	image

SAP IQ data type	Adaptive Server Enterprise default data type
TINYINT	tinyint
SMALLINT	smallint
INT, INTEGER	int
BIGINT	numeric(20,0)
UNSIGNED TINYINT	tinyint
UNSIGNED SMALLINT	int
UNSIGNED INTEGER	numeric(11,0)
UNSIGNED BIGINT	numeric(20,0)
CHAR(<n>)	if (n <= 255) char(<n>) else text
VARCHAR(<n>)	if (n <= 255) varchar(<n>) else text
LONG VARCHAR	text
NCHAR(<n>)	if (n <= 255) nchar(<n>) else ntext
NVARCHAR(<n>)	if (n <= 255) nvarchar(<n>) else ntext
LONG NVARCHAR	ntext
BINARY(<n>)	if (n <= 255) binary(<n>) else image
VARBINARY(<n>)	if (n <= 255) varbinary(<n>) else image
LONG BINARY	image
DECIMAL(<prec>,<scale>)	decimal(<prec>,<scale>)
NUMERIC(<prec>,<scale>)	numeric(<prec>,<scale>)
SMALLMONEY	numeric(10,4)
MONEY	numeric(19,4)
REAL	real
DOUBLE	float
FLOAT(<n>)	float(<n>)
DATE	datetime
TIME	datetime
SMALLDATETIME	smalldatetime
TIMESTAMP	datetime
TIMESTAMP WITH TIME ZONE	varchar(254)
XML	text
ST_GEOMETRY	image
UNIQUEIDENTIFIER	binary(16)

Example

Supply a connection string in the USING clause of the CREATE SERVER statement to connect to an Adaptive Server Enterprise 16 database.

```
CREATE SERVER TestASE
CLASS 'ASEODBC'
USING 'DRIVER=SAP ASE ODBC
Driver;Server=TestASE;Port=5000;Database=testdb;UID=username;PWD=password'
```

The driver name for Adaptive Server Enterprise 12 or earlier is **Sybase ASE ODBC Driver**.

The driver name for Adaptive Server Enterprise 15 is **Adaptive Server Enterprise**.

14.1.2.1.6 Server class DB2ODBC

A remote server with server class DB2ODBC is an IBM DB2 database server.

Notes

- SAP certifies the use of IBM's DB2 Connect version 5, with fix pack WR09044. Configure and test your ODBC configuration using the instructions for that product. SAP IQ has no specific requirements for the configuration of IBM DB2 data sources.
- The following is an example of a CREATE EXISTING TABLE statement for an IBM DB2 server with an ODBC data source named mydb2:

  ```
  CREATE EXISTING TABLE ibmcol
  AT 'mydb2..sysibm.syscolumns';
  ```

Data type conversions: IBM DB2

When you execute a CREATE TABLE statement, SAP IQ automatically converts the data types to the corresponding IBM DB2 data types.

The following table describes the SAP IQ to IBM DB2 data type conversions.

SAP IQ data type	IBM DB2 default data type
BIT	smallint
VARBIT(<n>)	if (n <= 4000) varchar(<n>) for bit data else long varchar for bit data
LONG VARBIT	long varchar for bit data
TINYINT	smallint

SAP IQ data type	IBM DB2 default data type
SMALLINT	smallint
INTEGER	int
BIGINT	decimal(20,0)
UNSIGNED TINYINT	int
UNSIGNED SMALLINT	int
UNSIGNED INTEGER	decimal(11,0)
UNSIGNED BIGINT	decimal(20,0)
CHAR(<n>)	if (n < 255) char(<n>) else if (n <= 4000) varchar(<n>) else long varchar
VARCHAR(<n>)	if (n <= 4000) varchar(<n>) else long varchar
LONG VARCHAR	long varchar
NCHAR(<n>)	Not supported
NVARCHAR(<n>)	Not supported
LONG NVARCHAR	Not supported
BINARY(<n>)	if (n <= 4000) varchar(<n>) for bit data else long varchar for bit data
VARBINARY(<n>)	if (n <= 4000) varchar(<n>) for bit data else long varchar for bit data
LONG BINARY	long varchar for bit data
DECIMAL(<prec>,<scale>)	decimal(<prec>,<scale>)
NUMERIC(<prec>,<scale>)	decimal(<prec>,<scale>)
SMALLMONEY	decimal(10,4)
MONEY	decimal(19,4)
REAL	real
DOUBLE	float
FLOAT(<n>)	float(<n>)
DATE	date
TIME	time
TIMESTAMP	timestamp
TIMESTAMP WITH TIME ZONE	varchar(254)
XML	long varchar for bit data
ST_GEOMETRY	long varchar for bit data
UNIQUEIDENTIFIER	varchar(16) for bit data

14.1.2.1.7 Server class HANAODBC

A remote server with server class HANAODBC is an SAP HANA database server.

Notes

- The following is an example of a CREATE EXISTING TABLE statement for an SAP HANA database server with an ODBC data source named mySAPHANA:

```
CREATE EXISTING TABLE hanatable
AT 'mySAPHANA..dbo.hanatable';
```

Data type conversions: SAP HANA

When you execute a CREATE TABLE statement, SAP IQ automatically converts the data types to the corresponding SAP HANA data types. The following table describes the SAP IQ to SAP HANA data type conversions.

SAP IQ data type	SAP HANA default data type
BIT	TINYINT
VARBIT(<n>)	if (n <= 5000) VARBINARY(<n>) else BLOB
LONG VARBIT	BLOB
TINYINT	TINYINT
SMALLINT	SMALLINT
INTEGER	INTEGER
BIGINT	BIGINT
UNSIGNED TINYINT	TINYINT
UNSIGNED SMALLINT	INTEGER
UNSIGNED INTEGER	BIGINT
UNSIGNED BIGINT	DECIMAL(20,0)
CHAR(<n>)	if (n <= 5000) VARCHAR(<n>) else CLOB
VARCHAR(<n>	if (n <= 5000) VARCHAR(<n>) else CLOB
LONG VARCHAR	CLOB
NCHAR(<n>)	if (n <= 5000) NVARCHAR(<n>) else NCLOB
NVARCHAR(<n>)	if (n <= 5000) NVARCHAR(<n>) else NCLOB
LONG NVARCHAR	NCLOB
BINARY(<n>)	if (n <= 5000) VARBINARY(<n>) else BLOB

SAP IQ data type	SAP HANA default data type
VARBINARY(<n>)	if (n <= 5000) VARBINARY(<n>) else BLOB
LONG BINARY	BLOB
DECIMAL(<precision>, <scale>)	DECIMAL(<precision>, <scale>)
NUMERIC(<precision>, <scale>)	DECIMAL(<precision>, <scale>)
SMALLMONEY	DECIMAL(13,4)
MONEY	DECIMAL(19,4)
REAL	REAL
DOUBLE	FLOAT
FLOAT(<n>)	FLOAT
DATE	DATE
TIME	TIME
TIMESTAMP	TIMESTAMP
TIMESTAMP WITH TIME ZONE	VARCHAR(254)
XML	BLOB
ST_GEOMETRY	BLOB
UNIQUEIDENTIFIER	VARBINARY(16)

14.1.2.1.8 Server class IQODBC

A remote server with server class IQODBC is an SAP IQ database server.

No special requirements exist for the configuration of an SAP IQ data source.

To access SAP IQ database servers that support multiple databases, create an ODBC data source name defining a connection to each database. Execute a CREATE SERVER statement for each of these ODBC data source names.

14.1.2.1.9 Server class MSACCESSODBC

Microsoft Access databases are stored in a `.mdb` file. Using the ODBC manager, create an ODBC data source and map it to one of these files.

A new `.mdb` file can be created through the ODBC manager. This database file becomes the default if you don't specify a different default when you create a table through SAP IQ.

Assuming an ODBC data source named access, you can use any of the following statements to access data:

- ```
  CREATE TABLE tab1 (a int, b char(10))
  AT 'access...tab1';
  ```
  ```

- ```
  CREATE TABLE tab1 (a int, b char(10))
  AT 'access;d:\\pcdb\\data.mdb;;tab1';
  ```
- ```
  CREATE EXISTING TABLE tab1
  AT 'access;d:\\pcdb\\data.mdb;;tab1';
  ```

Microsoft Access does not support the owner name qualification; leave it empty.

Data type conversions: Microsoft Access

SAP IQ data type	Microsoft Access default data type
BIT	TINYINT
VARBIT(<n>)	if (n <= 4000) BINARY(<n>) else IMAGE
LONG VARBIT	IMAGE
TINYINT	TINYINT
SMALLINT	SMALLINT
INTEGER	INTEGER
BIGINT	DECIMAL(19,0)
UNSIGNED TINYINT	TINYINT
UNSIGNED SMALLINT	INTEGER
UNSIGNED INTEGER	DECIMAL(11,0)
UNSIGNED BIGINT	DECIMAL(20,0)
CHAR(<n>)	if (n < 255) CHARACTER(<n>) else TEXT
VARCHAR(<n>)	if (n < 255) CHARACTER(<n>) else TEXT
LONG VARCHAR	TEXT
NCHAR(<n>)	Not supported
NVARCHAR(<n>)	Not supported
LONG NVARCHAR	Not supported
BINARY(<n>)	if (n <= 4000) BINARY(<n>) else IMAGE
VARBINARY(<n>)	if (n <= 4000) BINARY(<n>) else IMAGE
LONG BINARY	IMAGE
DECIMAL(<precision>, <scale>)	DECIMAL(<precision>, <scale>)
NUMERIC(<precision>, <scale>)	DECIMAL(<precision>, <scale>)
SMALLMONEY	MONEY
MONEY	MONEY
REAL	REAL
DOUBLE	FLOAT

SAP IQ data type	Microsoft Access default data type
FLOAT(<n>)	FLOAT
DATE	DATETIME
TIME	DATETIME
TIMESTAMP	DATETIME
TIMESTAMP WITH TIME ZONE	CHARACTER(254)
XML	XML
ST_GEOMETRY	IMAGE
UNIQUEIDENTIFIER	BINARY(16)

14.1.2.1.10 Server class MSSODBC

The server class MSSODBC is used to access Microsoft SQL Server through one of its ODBC drivers.

Notes

- Versions of Microsoft SQL Server ODBC drivers that have been used are:
  - Microsoft SQL Server ODBC Driver Version 06.01.7601
  - Microsoft SQL Server Native Client Version 10.00.1600
- The following is an example for Microsoft SQL Server:

```
CREATE SERVER mysqlserver
CLASS 'MSSODBC'
USING 'DSN=MSSODBC_cli';
CREATE EXISTING TABLE accounts
AT 'mysqlserver.master.dbo.accounts';
```

- The local setting of the quoted_identifier option controls the use of quoted identifiers for Microsoft SQL Server. For example, if you set the quoted_identifier option to Off locally, then quoted identifiers are turned off for Microsoft SQL Server.

Data type conversions: Microsoft SQL Server

When you execute a CREATE TABLE statement, SAP IQ automatically converts the data types to the corresponding Microsoft SQL Server data types using the following data type conversions.

SAP IQ data type	Microsoft SQL Server default data type
BIT	bit
VARBIT(<n>)	if (n <= 255) varbinary(<n>) else image

SAP IQ data type	Microsoft SQL Server default data type
LONG VARBIT	image
TINYINT	tinyint
SMALLINT	smallint
INTEGER	int
BIGINT	numeric(20,0)
UNSIGNED TINYINT	tinyint
UNSIGNED SMALLINT	int
UNSIGNED INTEGER	numeric(11,0)
UNSIGNED BIGINT	numeric(20,0)
CHAR(<n>)	if (n <= 255) char(<n>) else text
VARCHAR(<n>)	if (n <= 255) varchar(<n>) else text
LONG VARCHAR	text
NCHAR(<n>)	if (n <= 4000) nchar(<n>) else ntext
NVARCHAR(<n>)	if (n <= 4000) nvarchar(<n>) else ntext
LONG NVARCHAR	ntext
BINARY(<n>)	if (n <= 255) binary(<n>) else image
VARBINARY(<n>)	if (n <= 255) varbinary(<n>) else image
LONG BINARY	image
DECIMAL(<precision>, <scale>)	decimal(<precision>, <scale>)
NUMERIC(<precision>, <scale>)	numeric(<precision>, <scale>)
SMALLMONEY	smallmoney
MONEY	money
REAL	real
DOUBLE	float
FLOAT(<n>)	float(<n>)
DATE	datetime
TIME	datetime
SMALLDATETIME	smalldatetime
DATETIME	datetime
TIMESTAMP	datetime
TIMESTAMP WITH TIME ZONE	varchar(254)
XML	xml
ST_GEOMETRY	image
UNIQUEIDENTIFIER	binary(16)

14.1.2.1.11 Server class MYSQLODBC

When you execute a CREATE TABLE statement, SAP IQ automatically converts the data types to the corresponding Oracle MySQL data types.

SAP IQ data type	MySQL default data type
BIT	bit(1)
VARBIT(<n>)	if (n <= 4000) varbinary(<n>) else longblob
LONG VARBIT	longblob
TINYINT	tinyint unsigned
SMALLINT	smallint
INTEGER	int
BIGINT	bigint
UNSIGNED TINYINT	tinyint unsigned
UNSIGNED SMALLINT	int
UNSIGNED INTEGER	bigint
UNSIGNED BIGINT	decimal(20,0)
CHAR(<n>)	if (n < 255) char(<n>) else if (n <= 4000) varchar(<n>) else longtext
VARCHAR(<n>)	if (n <= 4000) varchar(<n>) else longtext
LONG VARCHAR	longtext
NCHAR(<n>)	if (n < 255) national character(<n>) else if (n <= 4000) national character varying(<n>) else longtext
NVARCHAR(<n>)	if (n <= 4000) national character varying(<n>) else longtext
LONG NVARCHAR	longtext
BINARY(<n>)	if (n <= 4000) varbinary(<n>) else longblob
VARBINARY(<n>)	if (n <= 4000) varbinary(<n>) else longblob
LONG BINARY	longblob
DECIMAL(<precision>, <scale>)	decimal(<precision>, <scale>)
NUMERIC(<precision>, <scale>)	decimal(<precision>, <scale>)

SAP IQ data type	MySQL default data type
SMALLMONEY	decimal(10,4)
MONEY	decimal(19,4)
REAL	real
DOUBLE	float
FLOAT(<n>)	float(<n>)
DATE	date
TIME	time
TIMESTAMP	datetime
TIMESTAMP WITH TIME ZONE	varchar(254)
XML	longblob
ST_GEOMETRY	longblob
UNIQUEIDENTIFIER	varbinary(16)

Example

Supply a connection string in the USING clause of the CREATE SERVER statement to connect to an Oracle MySQL database.

```
CREATE SERVER TestMySQL
CLASS 'MYSQLODBC'
USING 'DRIVER=MySQL ODBC 5.1
Driver;DATABASE=mydatabase;SERVER=mySQLHost;UID=me;PWD=secret'
```

14.1.2.1.12 Server class ODBC

ODBC data sources that do not have their own server class use ODBC server class.

You can use any ODBC driver. SAP certifies the following ODBC data sources:

- Microsoft Excel (Microsoft 3.51.171300)
- Microsoft Visual FoxPro (Microsoft 3.51.171300)
- Lotus Notes SQL

- Microsoft Excel (Microsoft 3.51.171300)
- Microsoft Visual FoxPro (Microsoft 3.51.171300)
- Lotus Notes SQL

The latest versions of Microsoft ODBC drivers can be obtained through the Microsoft Data Access Components (MDAC) distribution found at the Microsoft Download Center. The Microsoft driver versions listed above are part of MDAC 2.0.

In this section:

Microsoft Excel (Microsoft 3.51.171300) [page 322]
With Microsoft Excel, each Microsoft Excel workbook is logically considered to be a database holding several tables.

Microsoft Visual FoxPro (Microsoft 3.51.171300) [page 323]
You can store Microsoft Visual FoxPro tables together inside a single Microsoft Visual FoxPro database file (`.dbc`), or, you can store each table in its own separate `.dbf` file.

Lotus Notes SQL [page 323]
You can easily map SAP IQ tables to Notes forms and set up SAP IQ to access your Lotus Notes contacts.

14.1.2.1.12.1 Microsoft Excel (Microsoft 3.51.171300)

With Microsoft Excel, each Microsoft Excel workbook is logically considered to be a database holding several tables.

Tables are mapped to sheets in a workbook. When you configure an ODBC data source name in the ODBC driver manager, you specify a default workbook name associated with that data source. However, when you execute a CREATE TABLE statement, you can override the default and specify a workbook name in the location string. This allows you to use a single ODBC DSN to access all of your Microsoft Excel workbooks.

Create a remote server named excel that connects to the Microsoft Excel ODBC driver.

```
CREATE SERVER excel
CLASS 'ODBC'
USING 'DRIVER=Microsoft Excel Driver (*.xls);DBQ=d:\
\work1.xls;READONLY=0;DriverID=790'
```

To create a workbook named `work1.xls` with a sheet (table) called mywork:

```
CREATE TABLE mywork (a int, b char(20))
AT 'excel;d:\\work1.xls;;mywork';
```

To create a second sheet (or table) execute a statement such as:

```
CREATE TABLE mywork2 (x float, y int)
AT 'excel;d:\\work1.xls;;mywork2';
```

You can import existing sheets into SAP IQ using CREATE EXISTING, under the assumption that the first row of your sheet contains column names.

```
CREATE EXISTING TABLE mywork
AT'excel;d:\\work1;;mywork';
```

If SAP IQ reports that the table is not found, you may need to explicitly state the column and row range you want to map to. For example:

```
CREATE EXISTING TABLE mywork
AT 'excel;d:\\work1;;mywork$';
```

Adding the $ to the sheet name indicates that the entire worksheet should be selected.

Note in the location string specified by AT that a semicolon is used instead of a period for field separators. This is because periods occur in the file names. Microsoft Excel does not support the owner name field so leave this blank.

Deletes are not supported. Also some updates may not be possible since the Microsoft Excel driver does not support positioned updates.

Example

The following statements create a database server called TestExcel that uses an ODBC DSN to access the Microsoft Excel workbook `LogFile.xlsx` and import its sheet it into SAP IQ.

```
CREATE SERVER TestExcel
CLASS 'ODBC'
USING 'DRIVER=Microsoft Excel Driver (*.xls);DBQ=c:\\temp\
\LogFile.xlsx;READONLY=0;DriverID=790'
CREATE EXISTING TABLE MyWorkbook
AT 'TestExcel;c:\\temp\\LogFile.xlsx;;Logfile$';
SELECT * FROM MyWorkbook;
```

14.1.2.1.12.2 Microsoft Visual FoxPro (Microsoft 3.51.171300)

You can store Microsoft Visual FoxPro tables together inside a single Microsoft Visual FoxPro database file (`.dbc`), or, you can store each table in its own separate `.dbf` file.

When using `.dbf` files, be sure the file name is filled into the location string; otherwise the directory that SAP IQ was started in is used.

```
CREATE TABLE fox1 (a int, b char(20))
AT 'foxpro;d:\\pcdb;;fox1';
```

This statement creates a file named `d:\pcdb\fox1.dbf` when you choose the **Free Table Directory** option in the ODBC Driver Manager.

14.1.2.1.12.3 Lotus Notes SQL

You can easily map SAP IQ tables to Notes forms and set up SAP IQ to access your Lotus Notes contacts.

Prerequisites

You must obtain the Lotus Notes SQL driver.

Procedure

1. Make sure that the Lotus Notes program folder is in your path (for example, `C:\Program Files (x86)\IBM\Lotus\Notes`).
2. Create a 32-bit ODBC data source using the NotesSQL ODBC driver. Use the `names.nsf` database for this example. The **Map Special Characters** option should be turned on. For this example, the **Data Source Name** is `my_notes_dsn`.
3. Create a remote data access server using Interactive SQL connected to a 32-bit database server.

Results

You have set up SAP IQ to access your Lotus Notes contacts.

Example

- Create a remote data access server.

```
CREATE SERVER NotesContacts
CLASS 'ODBC'
USING 'my_notes_dsn';
```

- Create an external login for the Lotus Notes server.

```
CREATE EXTERNLOGIN "DBA" TO "NotesContacts"
REMOTE LOGIN 'John Doe/SAP' IDENTIFIED BY 'MyNotesPassword';
```

- Map some columns of the Person form into a SAP IQ table.

```
CREATE EXISTING TABLE PersonDetails
( DisplayName CHAR(254),
  DisplayMailAddress CHAR(254),
  JobTitle CHAR(254),
  CompanyName CHAR(254),
  Department CHAR(254),
  Location CHAR(254),
  OfficePhoneNumber CHAR(254) )
AT 'NotesContacts...Person';
```

- Query the table.

```
SELECT * FROM PersonDetails
WHERE Location LIKE 'Waterloo%';
```

14.1.2.1.13 Server class ORAODBC

A remote server with server class ORAODBC is an Oracle Database version 8.0 or later.

Notes

- SAP certifies the use of the Oracle Database version 8.0.03 ODBC driver. Configure and test your ODBC configuration using the instructions for that product.
- The following is an example of a CREATE EXISTING TABLE statement for an Oracle Database server named myora:

```
CREATE EXISTING TABLE employees
AT 'myora.database.owner.employees';
```

Data type conversions: Oracle Database

When you execute a CREATE TABLE statement, SAP IQ automatically converts the data types to the corresponding Oracle Database data types using the following data type conversions.

SAP IQ data type	Oracle Database data type
BIT	number(1,0)
VARBIT(<n>)	if (n <= 255) raw(<n>) else long raw
LONG VARBIT	long raw
TINYINT	number(3,0)
SMALLINT	number(5,0)
INTEGER	number(11,0)
BIGINT	number(20,0)
UNSIGNED TINYINT	number(3,0)
UNSIGNED SMALLINT	number(5,0)
UNSIGNED INTEGER	number(11,0)
UNSIGNED BIGINT	number(20,0)
CHAR(<n>)	if (n <= 255) char(<n>) else long
VARCHAR(<n>)	if (n <= 2000) varchar(<n>) else long
LONG VARCHAR	long

SAP IQ data type	Oracle Database data type
NCHAR(<n>)	if (n <= 255) nchar(<n>) else nclob
NVARCHAR(<n>)	if (n <= 2000) nvarchar(<n>) else nclob
LONG NVARCHAR	nclob
BINARY(<n>)	if (n > 255) long raw else raw(<n>)
VARBINARY(<n>)	if (n > 255) long raw else raw(<n>)
LONG BINARY	long raw
DECIMAL(<precision>, <scale>)	number(<precision>, <scale>)
NUMERIC(<precision>, <scale>)	number(<precision>, <scale>)
SMALLMONEY	numeric(13,4)
MONEY	number(19,4)
REAL	real
DOUBLE	float
FLOAT(<n>)	float
DATE	date
TIME	date
TIMESTAMP	date
TIMESTAMP WITH TIME ZONE	varchar(254)
XML	long raw
ST_GEOMETRY	long raw
UNIQUEIDENTIFIER	raw(16)

Example

Supply a connection string in the USING clause of the CREATE SERVER statement to connect to an Oracle database.

```
CREATE SERVER TestOracle
CLASS 'ORAODBC'
USING 'DRIVER=Oracle ODBC Driver;DBQ=mydatabase;UID=username;PWD=password'
```

14.1.3 Remote Servers

Before remote objects can be mapped to a local proxy table, define the remote server where the remote object is located.

In this section:

Create Remote Servers [page 327]
Use the `CREATE SERVER` statement to set up remote server definitions.

Before You Access Remote Oracle Data [page 327]
To access remote Oracle data, configure your system with the prerequisite software.

Connecting to an Oracle Database [page 330]
Connect SAP IQ to remote Oracle data via Component Integration Services.

Troubleshoot Oracle Database Access [page 331]
If Oracle data access returns errors, check the appropriate configuration component.

Loading Remote Data Without Native Classes [page 332]
Load data by using DirectConnect.

Querying Data Without Native Classes [page 333]
Follow these guidelines to query data without native classes.

Delete Remote Servers [page 336]
Use the `DROP SERVER` statement to delete a remote server from the `ISYSSERVER` system table.

Alter Remote Servers [page 337]
Use the `ALTER SERVER` statement to modify the attributes of a server. These changes do not take effect until the next connection to the remote server.

Listing the tables on a remote server (SQL) [page 337]
View a limited or comprehensive list of all the tables on a remote server using a system procedure.

Remote server capabilities [page 338]
The database server uses remote server capability information to determine how much of a SQL statement can be passed to a remote server.

14.1.3.1 Create Remote Servers

Use the `CREATE SERVER` statement to set up remote server definitions.

For some systems, including SAP IQ and SAP SQL Anywhere, each data source describes a database, so a separate remote server definition is needed for each database.

14.1.3.2 Before You Access Remote Oracle Data

To access remote Oracle data, configure your system with the prerequisite software.

1. Check for Prerequisites [page 328]
   Check your system for the software components required to access Oracle data using Component Integration Services (CIS).
2. Create an Oracle Data Source Name [page 328]
   Use the `iqdsn` utility to create an entry in the `.odbc.ini` file.
3. Set Environment Variables for Oracle Data Access [page 329]
   Before starting the SAP IQ server to access Oracle data, you must set certain environment variables.
4. Start the SAP IQ Server [page 330]
   Start the SAP IQ server that you will use as a front end to access Oracle data.

14.1.3.2.1 Check for Prerequisites

Check your system for the software components required to access Oracle data using Component Integration Services (CIS).

Prerequisites are:

- An Oracle database
- Oracle client software (basic package), including a `network/admin/tnsnames.ora` file.
- A platform-specific driver (installed with SAP IQ):
  - AIX 64 – `$IQDIR16/libxx/libdboraodbc12_r.so`
  - HP-UX – `$IQDIR16/libxx/libdboraodbc12_r.so.1`
  - Linux64 – `$IQDIR16/libxx/libdboraodbc12_r.so.1`
  - SunOS64 – `$IQDIR16/libxx/libdboraodbc12_r.so.1`
  - WinAMD64 – `%IQDIR16%\bin64\dboraodbc12.dll`

Parent topic: Before You Access Remote Oracle Data [page 327]

Next task: Create an Oracle Data Source Name [page 328]

14.1.3.2.2 Create an Oracle Data Source Name

Use the `iqdsn` utility to create an entry in the `.odbc.ini` file.

Procedure

1. Display Oracle connection keywords:

   ```
   % iqdsn -cl -or
   ```

   ```
   Driver
   UserID              UID
   Password            PWD
   ```

```
SID                 SID
Encrypted Password  ENP
ProcResults         PROC
ArraySize           SIZE
EnableMSDTC         EDTC
ProcOwner           POWNER
```

2. Create an `.odbc.ini` file entry:

```
% iqdsn -or -y -w "MyOra2" -c "UID=system;PWD=manager;SID=QAORA"
```

```
[MyOra2]
Driver=/Sybase/IQ-16_1/lib64/libdboraodbc12_r.so
UserID=system
Password=manager
SID=QAORA
```

Task overview: Before You Access Remote Oracle Data [page 327]

Previous: Check for Prerequisites [page 328]

Next: Set Environment Variables for Oracle Data Access [page 329]

14.1.3.2.3 Set Environment Variables for Oracle Data Access

Before starting the SAP IQ server to access Oracle data, you must set certain environment variables.

Set these variables for Oracle access:

- ORACLE_HOME:

```
setenv ORACLE_HOME
```

- ODBCINI, where `<location>` is the location of the .odbc.ini file in the Oracle Entry:

```
setenv ODBCINI <location>
```

- The library path variable for your platform:
  - AIX, where `<path>` is the path to the platform-specific Oracle client directory:

```
setenv LIBPATH <path> $LIBPATH
```

  - Other UNIX platforms, where `<path>` is the path to the platform-specific Oracle client directory:

```
setenv LD_LIBRARY_PATH <path>;$LD_LIBRARY_PATH
```

Parent topic: Before You Access Remote Oracle Data [page 327]

Previous task: Create an Oracle Data Source Name [page 328]

Next task: Start the SAP IQ Server [page 330]

14.1.3.2.4 Start the SAP IQ Server

Start the SAP IQ server that you will use as a front end to access Oracle data.

Procedure

Use the `-n` database option in the `start_iq` to specify the name of the server you are starting.

This example starts the server named myserver:

```
start_iq -n myserver
```

Task overview: Before You Access Remote Oracle Data [page 327]

Previous: Set Environment Variables for Oracle Data Access [page 329]

14.1.3.3 Connecting to an Oracle Database

Connect SAP IQ to remote Oracle data via Component Integration Services.

Prerequisites

Log in to `dbisql` or `iqisql`.

Procedure

1. Create a server using the data source name from the `.odbc.ini` file:
   ```
   CREATE SERVER myora CLASS 'oraodbc' USING 'MyOra2'
   ```
2. Create an external login:
   ```
   CREATE EXTERNLOGIN DBA TO myora REMOTE LOGIN system IDENTIFIED BY manager
   ```
3. Confirm the connection:
   ```
   sp_remote_tables myora
   ```
4. Create a table of Oracle data:
   ```
   CREATE EXISTING TABLE my_oratable at 'myora..system.oratable'
   ```
5. Verify that the connection works by selecting data:
   ```
   SELECT * FROM my_oratable
   ```

14.1.3.4 Troubleshoot Oracle Database Access

If Oracle data access returns errors, check the appropriate configuration component.

1. Error Loading Driver [page 331]
   A driver load error may indicate a problem with an environment variable or the configuration information file.
2. Error Resolving Connect Identifier [page 331]
   An error resolving the connect identifier may be a problem with the Oracle definition, an environment variable, or the configuration information file.

14.1.3.4.1 Error Loading Driver

A driver load error may indicate a problem with an environment variable or the configuration information file.

Peform the following if you see an error message such as `Can't load driver`:

- Check that the `.odbc.ini` entry lists the correct driver.
- Check that the Oracle client software is added to the LD_LIBRARY_PATH definition.

Parent topic: Troubleshoot Oracle Database Access [page 331]

Next: Error Resolving Connect Identifier [page 331]

14.1.3.4.2 Error Resolving Connect Identifier

An error resolving the connect identifier may be a problem with the Oracle definition, an environment variable, or the configuration information file.

If you see the error `ORA-12154: TNS:could not resolve the connect identifier`, perform the following:

- Check that the Oracle definition is correct.
- Check that ORACLE_HOME is set correctly.
- Check that the gateway system identifier (SID) entered in `.odbc.ini` is correct.

Parent topic: Troubleshoot Oracle Database Access [page 331]

Previous: Error Loading Driver [page 331]

14.1.3.5 Loading Remote Data Without Native Classes

Load data by using DirectConnect.

Context

Native classes use DirectConnect to access remote data sources:

- On 64-bit UNIX platforms
- On 32-bit platforms where no ODBC driver is available (for example, Microsoft SQL Server)

In this section:

Loading MS SQL Server Data into an SAP IQ Server on UNIX [page 332]
This remote data example loads MS SQL Server data into an SAP IQ server on UNIX.

14.1.3.5.1 Loading MS SQL Server Data into an SAP IQ Server on UNIX

This remote data example loads MS SQL Server data into an SAP IQ server on UNIX.

Context

For this example, assume the following:

- An Enterprise Connect Data Access (ECDA) server named `mssql` exists on UNIX host `myhostname`, port 12530.
- The data is to be retrieved from an MS SQL server named `2000` on host `myhostname`, port 1433.

Procedure

1. Using DirectConnect documentation, configure DirectConnect for your data source.
2. Verify that ECDA server (`<mssql>`) is listed in the SAP IQ interfaces file:

```
mssql
master tcp ether myhostname 12530
query tcp ether myhostname 12530
```

3. Add a new user, using the user ID and password for server `<mssql>`:

```
isql -Udba -P<password> -Stst_iqdemo
```

```
grant connect to chill identified by chill
grant dba to chill
```

4. Log in as the new user to create a local table on SAP IQ:

```
isql -Uchill -P<password> -Stst_iqdemo
create table billing(status char(1), name varchar(20), telno int)
```

5. Insert data:

```
insert into billing location 'mssql.pubs' { select * from billing }
```

14.1.3.6 Querying Data Without Native Classes

Follow these guidelines to query data without native classes.

Procedure

1. Configure ASE/CIS with a remote server and proxy to connect via DirectConnect. For example, use DirectConnect for Oracle to the Oracle server.
2. Configure SAP IQ with a remote server using the ASEODBC class to the SAP Adaptive Server Enterprise server.
3. Use the `CREATE EXISTING TABLE` statement to create proxy tables pointing to the proxy tables in ASE, which in turn point to Oracle.

In this section:

Querying Remote Data Using DirectConnect and Proxy Table from UNIX [page 333]
Query data using DirectConnect.

14.1.3.6.1 Querying Remote Data Using DirectConnect and Proxy Table from UNIX

Query data using DirectConnect.

This example shows how to access MS SQL Server data. For this example, assume the following:

- An SAP IQ server on host `myhostname`, port 7594.
- An SAP Adaptive Server Enterprise server on host `myhostname`, port 4101.
- An Enterprise Connect Data Access (ECDA) server exists named `mssql` on host `myhostname`, port 12530.
- The data is to be retrieved from an MS SQL server named `2000` on host `myhostname`, port 1433.

In this section:

Setting Up SAP ASE to Query MS SQL Server [page 334]

Set up SAP Adaptive Server Enterprise and Component Integration Services (CIS) to query MS SQL Server through DirectConnect.

Setting up SAP IQ to Connect to the SAP ASE Server [page 336]
Follow these steps to query SAP Adaptive Server Enterprise data.

14.1.3.6.1.1 Setting Up SAP ASE to Query MS SQL Server

Set up SAP Adaptive Server Enterprise and Component Integration Services (CIS) to query MS SQL Server through DirectConnect.

Context

For this example, assume that the server name is <jones_1207>.

Procedure

1. Add an entry to the SAP Adaptive Server Enterprise interfaces file to connect to <mssql>:

```
mssql
```

```
master tcp ether hostname 12530
```

```
query tcp ether hostname 12530
```

2. Enable CIS and remote procedure call handling from the SAP ASE server. For example, if CIS is already enabled as the default:

```
sp_configure 'enable cis'
```

```
Parameter Name Default Memory Used Config Value Run Value
```

```
enable cis                          1         0                 1              1
```

```
(1 row affected)
(return status=0)
```

```
sp_configure 'cis rpc handling', 1
```

```
Parameter Name Default Memory Used Config Value Run Value
```

```
enable cis                          0         0                 0              1
```

```
(1 row affected)
```

```
Configuration option changed. The SQL Server need not be restarted since the
option is dynamic.
```

3. Add the DirectConnect server to the SAP ASE server's `SYSSERVERS` system table.

```
sp_addserver mssql, direct_connect, mssql
```

```
Adding server 'mssql', physical name 'mssql'
Server added.
(Return status=0)
```

4. Create the user in SAP ASE that will be used in SAP IQ to connect to SAP ASE.

```
sp_addlogin tst, tsttst
```

```
Password correctly set.
Account unlocked. New login created.
(return status = 0)
```

```
grant role sa_role to tst
use tst_db
sp_adduser tst
```

```
New user added.
(return status = 0)
```

5. Add an external login from the master database:

```
use master
sp_addexternlogin mssql, tst, chill, chill
```

```
User 'tst' will be known as 'chill' in remote server 'mssql'.
(return status = 0)
```

6. Create an ASE proxy table as the added user from the desired database:

```
isql -Utst -Ttsttst
use test_db
create proxy_table billing_tst at 'mssql.pubs..billing'
select * from billing_tst
```

```
status    name          telno
------    -----------   -----
D         BOTANICALLY   1
B         BOTANICALL    2
(2 rows affected)
```

14.1.3.6.1.2 Setting up SAP IQ to Connect to the SAP ASE Server

Follow these steps to query SAP Adaptive Server Enterprise data.

Procedure

1. Add an entry to the SAP IQ interfaces file:

```
jones_1207
master tcp ether jones 4101
query tcp ether jones 4101
```

2. Create the user to connect to SAP Adaptive Server Enterprise:

```
GRANT CONNECT TO tst IDENTIFIED BY tsttst
GRANT dba TO tst
```

3. Log in as the added user to create the 'aseodbc' server class and add external login:

```
isql -Utst -Ptsttst -Stst_iqdemo
CREATE SERVER jones_1207 CLASS 'aseodbc' USING 'jones:4101/tst_db'
CREATE EXISTING TABLE billing_iq AT 'jones_1207.tst_db..billing_txt'
SELECT * from billing_iq
```

```
status               name          telno
------    -----------   -----
D         BOTANICALLY   1
B         BOTANICALL    2
(2 rows affected)
```

14.1.3.7 Delete Remote Servers

Use the `DROP SERVER` statement to delete a remote server from the `ISYSSERVER` system table.

Prerequisites

All remote tables defined on that server must already be dropped for this action to succeed.

Procedure

Execute the following DROP SERVER statement, specifying the name of the remote server.

This example drops the server named RemoteSA:

```
DROP SERVER RemoteSA;
```

14.1.3.8 Alter Remote Servers

Use the `ALTER SERVER` statement to modify the attributes of a server. These changes do not take effect until the next connection to the remote server.

Procedure

Execute an `ALTER SERVER` statement.

The following statement changes the server class of the server named RemoteASE to aseodbc. In this example, the Data Source Name for the server is RemoteASE:

```
ALTER SERVER RemoteASE
CLASS 'aseodbc';
```

14.1.3.9 Listing the tables on a remote server (SQL)

View a limited or comprehensive list of all the tables on a remote server using a system procedure.

Procedure

Call the sp_remote_tables system procedure to return a list of the tables on a remote server.

If you specify `<@table_name>` or `<@table_owner>`, the list of tables is limited to only those that match.

Results

A list of all the tables, or a limited list of tables, is returned.

Example

To get a list of all the tables in a database at the remote server named RemoteSA, owned by GROUPO, execute the following statement:

```
CALL sp_remote_tables('RemoteSA', null, 'GROUPO');
```

To get a list of all the tables in the Production database in an Adaptive Server Enterprise server named RemoteASE, owned by Fred, execute the following statement:

```
CALL sp_remote_tables('RemoteASE', null, 'Fred', 'Production');
```

To get a list of all the Microsoft Excel worksheets available from a remote server named Excel, execute the following statement:

```
CALL sp_remote_tables('Excel');
```

14.1.3.10 Remote server capabilities

The database server uses remote server capability information to determine how much of a SQL statement can be passed to a remote server.

Use the sp_servercaps system procedure to return the capabilities of a remote server.

You can also view capability information for remote servers by querying the SYSCAPABILITY and SYSCAPABILITYNAME system views. These system views are empty until after SAP IQ first connects to a remote server.

When using the sp_servercaps system procedure, the `<server-name>` specified must be the same `<server-name>` used in the CREATE SERVER statement.

Execute the stored procedure sp_servercaps as follows:

```
CALL sp_servercaps('<server-name>');
```

14.1.4 External Logins

SAP IQ uses the names and passwords of its clients when it connects to a remote server on behalf of those clients. However, this behavior can be overridden by creating external logins.

External logins are alternate login names and passwords that are used when communicating with a remote server.

When SAP IQ connects to the remote server, `INSERT...LOCATION` uses the remote login for the user ID of the current connection, if a remote login has been created with `CREATE EXTERNLOGIN` and the remote server has been defined with a `CREATE SERVER` statement.

If the remote server is not defined, or a remote login has not been created for the user ID of the current connection, SAP IQ connects using the user ID and password of the current connection.

> **Note**
>
> If you rely on the default user ID and password, and a user changes the password, you must stop and restart the server before the new password takes effect on the remote server. Remote logins created with `CREATE EXTERNLOGIN` are unaffected by changes to the password for the default user ID.

If you use an integrated login, the SAP IQ name and password of the SAP IQ client is the same as the database login ID and password that the SAP IQ userid maps to in `syslogins`.

14.1.5 Proxy tables

Use a proxy table to access any object (including tables, views, and materialized views) that the remote database exports as a candidate for a proxy table.

Proxy tables and remote servers

Location transparency of remote data is enabled by creating a local **proxy table** that maps to the remote object. Use one of the following statements to create a proxy table:

- If the table already exists at the remote storage location, use the CREATE EXISTING TABLE statement. This statement defines the proxy table for an existing table on the remote server.
- If the table does not exist at the remote storage location, use the CREATE TABLE statement. This statement creates a new table on the remote server, and also defines the proxy table for that table.

> **Note**
>
> You cannot modify data in a proxy table when you are within a savepoint.
>
> When a trigger is fired on a proxy table, the permissions used are those of the user who caused the trigger to fire, not those of the proxy table owner.

Proxy tables and directory access servers

A directory access server must have one and only one proxy table.

In this section:

Proxy table locations [page 340]
: Use the AT clause of the CREATE TABLE and the CREATE EXISTING TABLE statements to define the location of an existing object.

Creating proxy tables (SQL) [page 341]
: Create proxy tables with either the CREATE TABLE or CREATE EXISTING TABLE statement.

List the columns on a remote table [page 342]

Before you query a proxy table, it may be helpful to get a list of the columns that are available on a remote table.

14.1.5.1 Proxy table locations

Use the AT clause of the CREATE TABLE and the CREATE EXISTING TABLE statements to define the location of an existing object.

When you create a proxy table by using either the CREATE TABLE or the CREATE EXISTING statement, the AT clause includes a location string that is comprised of the following parts:

- The name of the remote server
- The remote catalog
- The remote owner or schema
- The remote table name
- (Optionally) an ESCAPE CHARACTER clause - but only for cases where an object name contain an unusual character such as period, semicolon, brace, and so on.

The syntax of the AT clause is:

```
... AT '<server>.<database>.<owner>.<table-name>' [ ESCAPE CHARACTER
<character> ]
```

server

Specifies the name by which the server is known in the current database, as specified in the CREATE SERVER statement. This field is mandatory for all remote data sources.

database

The meaning of the database field depends on the data source. Sometimes this field does not apply and should be left empty. The delimiter is still required, however.

If the data source is Adaptive Server Enterprise, then `<database>` specifies the database where the table exists. For example master or pubs2.

If the data source is SAP IQ, then this field does not apply; leave it empty.

If the data source is Microsoft Excel, Lotus Notes, or Microsoft Access, then include the name of the file containing the table. If the file name includes a period, then use the semicolon delimiter.

owner

If the database supports the concept of ownership, then this field represents the owner name. This field is only required when several owners have tables with the same name.

table-name

Specifies the name of the table. For a Microsoft Excel spreadsheet, this is the name of the sheet in the workbook. If `<table-name>` is left empty, then the remote table name is assumed to be the same as the local proxy table name.

string

Specifies the character to escape in a remote server name, catalog name, owner name, schema name, or table name. For example, if `<table-name>` contains a character such as period, semicolon, and a brace, it must be escaped by specifying the ESCAPE CHARACTER clause.

Example

The following examples illustrate the use of location strings:

- SAP IQ:

```
'RemoteSA..GROUPO.Employees'
```

- Adaptive Server Enterprise:

```
'RemoteASE.pubs2.dbo.publishers'
```

- Microsoft Excel:

```
'RemoteExcel;d:\pcdb\quarter3.xls;;sheet1$'
```

- Microsoft Access:

```
'RemoteAccessDB;\\server1\production\inventory.mdb;;parts'
```

14.1.5.2 Creating proxy tables (SQL)

Create proxy tables with either the CREATE TABLE or CREATE EXISTING TABLE statement.

Prerequisites

You must have the CREATE PROXY TABLE system privilege to create proxy tables owned by you. You must have the CREATE ANY TABLE or CREATE ANY OBJECT system privilege to create proxy tables owned by others.

Context

The CREATE TABLE statement creates a new table on the remote server, and defines the proxy table for that table when you use the AT clause. The AT clause specifies the location of the remote object, using periods or semicolons as delimiters. The ESCAPE CHARACTER clause allows applications to escape these delimiters within a location string. SAP IQ automatically converts the data into the remote server's native types.

If you use the CREATE TABLE statement to create both a local and remote table, and then subsequently use the DROP TABLE statement to drop the proxy table, the remote table is also dropped. Use the DROP TABLE statement to drop a proxy table created using the CREATE EXISTING TABLE statement however. In this case, the remote table is not dropped.

The CREATE EXISTING TABLE statement creates a proxy table that maps to an existing table on the remote server. The database server derives the column attributes and index information from the object at the remote location.

Procedure

1. Connect to the host database.
2. Execute a CREATE EXISTING TABLE statement.

Results

The proxy table is created.

Example

The following statement creates a proxy table called p_Employees on the current server that maps to a remote table named Employees on the server named RemoteSA, use the following statement:

```
CREATE EXISTING TABLE p_Employees
AT 'RemoteSA..GROUPO.Employees';
```

The following statement maps the proxy table a1 to the Microsoft Access file `mydbfile.mdb`. In this example, the AT clause uses the semicolon (;) as a delimiter. The server defined for Microsoft Access is named access.

```
CREATE EXISTING TABLE a1
AT 'access;d:\mydbfile.mdb;;a1';
```

The following statement creates a table named Employees on the remote server RemoteSA, and creates a proxy table named Members that maps to the remote table:

```
CREATE TABLE Members
( membership_id INTEGER NOT NULL,
member_name CHAR( 30 ) NOT NULL,
office_held CHAR( 20 ) NULL )
AT 'RemoteSA..GROUPO.Employees';
```

14.1.5.3 List the columns on a remote table

Before you query a proxy table, it may be helpful to get a list of the columns that are available on a remote table.

The sp_remote_columns system procedure produces a list of the columns on a remote table and a description of those data types. The following is the syntax for the sp_remote_columns system procedure:

```
CALL sp_remote_columns( @server_name, @table_name [, @table_owner [,
@table_qualifier ] ] )
```

If a table name, owner, or database name is given, the list of columns is limited to only those that match.

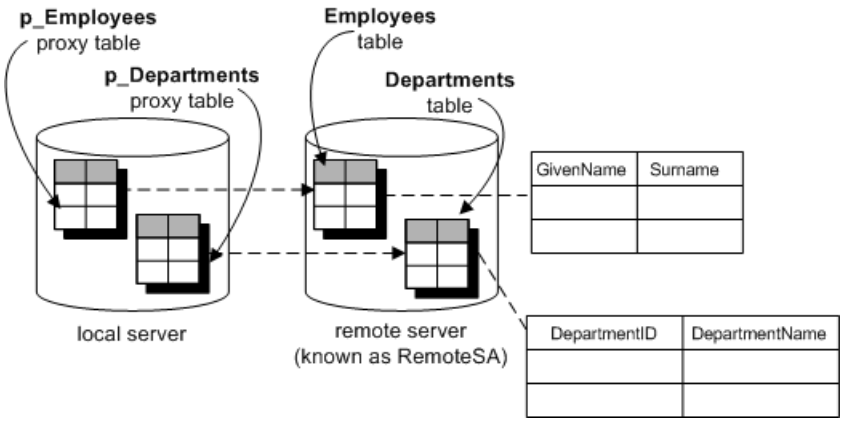
For example, the following returns a list of the columns in the sysobjects table in the production database on an Adaptive Server Enterprise server named asetest:

```
CALL sp_remote_columns('asetest, 'sysobjects', null, 'production');
```

14.1.6 Joins between remote tables

You can use joins between proxy tables and remote tables.

The following figure illustrates proxy tables on a local database server that are mapped to the remote tables Employees and Departments of the SAP IQ sample database on the remote server RemoteSA.



Example

The following example performs a join between two remote tables:

1. Create a new database named `empty.db`.
   This database holds no data. It is used only to define the remote objects, and to access the SAP IQ sample database.
2. Start a database server running the `empty.db`. You can do this by running the following command:

   ```
   start_iq empty
   ```
3. From Interactive SQL, connect to `empty.db` as user DBA.
4. In the new database, create a remote server named RemoteSA. Its server class is SAODBC, and the connection string refers to the SAP IQ 16 Demo ODBC data source:

   ```
   CREATE SERVER RemoteSA
   CLASS 'SAODBC'
   USING 'SAP IQ 16 Demo;PWD=sql';
   ```
5. In this example, you use the same user ID and password on the remote database as on the local database, so no external logins are needed.

Sometimes you must provide a user ID and password when connecting to the database at the remote server. In the new database, you could create an external login to the remote server. For simplicity in this example, the local login name and the remote user ID are both DBA:

```
CREATE EXTERNLOGIN DBA
TO RemoteSA
REMOTE LOGIN DBA
IDENTIFIED BY sql;
```

6. Define the p_Employees proxy table:

```
CREATE EXISTING TABLE p_Employees
AT 'RemoteSA..GROUPO.Employees';
```

7. Define the p_Departments proxy table:

```
CREATE EXISTING TABLE p_Departments
AT 'RemoteSA..GROUPO.Departments';
```

8. Use the proxy tables in the SELECT statement to perform the join.

```
SELECT GivenName, Surname, DepartmentName
FROM p_Employees JOIN p_Departments
ON p_Employees.DepartmentID = p_Departments.DepartmentID
ORDER BY Surname;
```

14.1.7 Joins between tables from multiple local databases

A database server may have several local databases running at one time. By defining tables in other local SAP IQ databases as remote tables, you can perform cross-database joins.

Example

Suppose you are using database db1, and you want to access data in tables in database db2. You need to set up proxy table definitions that point to the tables in database db2. For example, on a SAP IQ server named RemoteSA, you might have three databases available: db1, db2, and db3.

1. If you are using ODBC, create an ODBC data source name for each database you will be accessing.
2. Connect to the database from which you will be performing the join. For example, connect to db1.
3. Perform a CREATE SERVER statement for each other local database you will be accessing. This sets up a **loopback** connection to your SAP IQ server.

```
CREATE SERVER remote_db2
CLASS 'SAODBC'
USING 'RemoteSA_db2';
CREATE SERVER remote_db3
CLASS 'SAODBC'
USING 'RemoteSA_db3';
```

4. Create proxy table definitions by executing CREATE EXISTING TABLE statements for the tables in the other databases you want to access.

```
CREATE EXISTING TABLE Employees
```

```
AT 'remote_db2...Employees';
```

14.1.8 Native statements and remote servers

Use the FORWARD TO statement to send one or more statements to the remote server in its native syntax.

This statement can be used in two ways:

- To send a statement to a remote server.
- To place SAP IQ into passthrough mode for sending a series of statements to a remote server.

The FORWARD TO statement can be used to verify that a server is configured correctly. If you send a statement to the remote server and SAP IQ does not return an error message, the remote server is configured correctly.

The FORWARD TO statement cannot be used within procedures or batches.

If a connection cannot be made to the specified server, a message is returned to the user. If a connection is made, any results are converted into a form that can be recognized by the client program.

Example

Example 1

The following statement verifies connectivity to the server named RemoteASE by selecting the version string:

```
FORWARD TO RemoteASE {SELECT @@version};
```

Example 2

The following statements show a passthrough session with the server named RemoteASE:

```
FORWARD TO RemoteASE;
    SELECT * FROM titles;
    SELECT * FROM authors;
FORWARD TO;
```

14.1.9 Remote Procedure Calls (RPCs)

SAP IQ users can issue procedure calls to remote servers that support the feature.

SAP IQ, SAP SQL Anywhere, and SAP Adaptive Server Enterprise, as well as Oracle and DB2, support this feature. Issuing a remote procedure call is similar to using a local procedure call.

In this section:

Creating Remote Procedures [page 346]
Administrators can create remote procedures in Interactive SQL.

14.1.9.1 Creating Remote Procedures

Administrators can create remote procedures in Interactive SQL.

Prerequisites

You must have the MANAGE REPLICATION system privilege.

Context

If a remote procedure can return a result set, even if it does not always return one, then the local procedure definition must contain a RESULT clause.

Procedure

1. Connect to the host database.
2. Execute a statement to define the procedure.

   For example:

   ```
   CREATE PROCEDURE RemoteWho()
   AT 'bostonase.master.dbo.sp_who'
   ```

   This example specifies a parameter when calling a remote procedure:

   ```
   CREATE PROCEDURE RemoteUser ( IN username CHAR( 30 ) )
   AT 'bostonase.master.dbo.sp_helpuser';
   CALL RemoteUser( 'joe' );
   ```

14.2 Internal Operations

This section describes the underlying steps that SAP IQ performs on remote servers on behalf of client applications.

In this section:

Query Parsing [page 347]
When a statement is received from a client, the database server parses it. The database server raises an error if the statement is not a valid SQL Anywhere SQL statement.

Query Normalization [page 347]

In query normalization, referenced objects are verified and data type compatibility is checked.

Internal operations performed on queries [page 347]
The are several steps that are performed on all queries, both local and remote.

Complete passthrough of the statement [page 348]
For efficiency, SAP IQ passes off as much of the statement as possible to the remote server.

Partial passthrough of the statement [page 348]
If a statement contains references to multiple servers, or uses SQL features not supported by a remote server, the query is broken into simpler parts.

14.2.1 Query Parsing

When a statement is received from a client, the database server parses it. The database server raises an error if the statement is not a valid SQL Anywhere SQL statement.

14.2.2 Query Normalization

In query normalization, referenced objects are verified and data type compatibility is checked.

For example, consider this query:

```
SELECT *
FROM t1
WHERE c1 = 10
```

The query normalization stage verifies that table `t1` with a column `c1` exists in the system tables. It also verifies that the data type of column `c1` is compatible with the value 10. If the column's data type is `DATETIME`, for example, this statement is rejected.

14.2.3 Internal operations performed on queries

The are several steps that are performed on all queries, both local and remote.

Query preprocessing

Query preprocessing prepares the query for optimization. It may change the representation of a statement so that the SQL statement that SAP IQ generates for passing to a remote server is syntactically different from the original statement, even though it is semantically equivalent.

Preprocessing performs view expansion so that a query can operate on tables referenced by the view. Expressions may be reordered and subqueries may be transformed to improve processing efficiency. For example, some subqueries may be converted into joins.

14.2.4 Complete passthrough of the statement

For efficiency, SAP IQ passes off as much of the statement as possible to the remote server.

Often, this is the complete statement originally given to SAP IQ.

SAP IQ hands off the complete statement when:

- Every table in the statement resides on the same remote server.
- The remote server can process all of the syntax in the statement.

In rare conditions, it may actually be more efficient to let SAP IQ do some of the work instead of the remote server doing it. For example, SAP IQ may have a better sorting algorithm. In this case, you may consider altering the capabilities of a remote server using the ALTER SERVER statement.

14.2.5 Partial passthrough of the statement

If a statement contains references to multiple servers, or uses SQL features not supported by a remote server, the query is broken into simpler parts.

SELECT

SELECT statements are broken down by removing portions that cannot be passed on and letting SAP IQ perform the work. For example, suppose a remote server cannot process the ATAN2 function in the following statement:

```
SELECT a,b,c
WHERE ATAN2( b, 10 ) > 3
AND c = 10;
```

The statement sent to the remote server would be converted to:

```
SELECT a,b,c WHERE c = 10;
```

Then, SAP IQ locally applies `WHERE ATAN2( b, 10 ) > 3` to the intermediate result set.

Joins

When a statement contains joins between tables in multiple locations, IQ will attempt to push joins of collocated tables to the server on which they reside. The results of that join will then be joined by IQ with results from other remote tables or local tables. IQ will always prefer to push as much join work as is possible to remote servers. When IQ joins remote tables with local IQ tables, IQ may choose to use any join algorithm it supports.

The choice of algorithm is based on cost estimates. These algorithms can include nested loop, hash, or sort-merge joins.

When a nested loop join is chosen between an IQ and a remote table, every effort is made to make the remote table the outermost table in the join. This is due to the high cost of network I/O that makes look-ups against a remote table usually much higher than a local table.

UPDATE and DELETE

When a qualifying row is found, if SAP IQ cannot pass off an UPDATE or DELETE statement entirely to a remote server, it must change the statement into a table scan containing as much of the original WHERE clause as possible, followed by a positioned UPDATE or DELETE statement that specifies WHERE CURRENT OF <cursor-name>.

For example, when the function ATAN2 is not supported by a remote server:

```
UPDATE t1
SET a = atan2( b, 10 )
WHERE b > 5;
```

Would be converted to the following:

```
SELECT a,b
FROM t1
WHERE  b > 5;
```

Each time a row is found, SAP IQ would calculate the new value of a and execute:

```
UPDATE t1
SET a = 'new value'
WHERE CURRENT OF CURSOR;
```

If a already has a value that equals the new value, a positioned UPDATE would not be necessary, and would not be sent remotely.

To process an UPDATE or DELETE statement that requires a table scan, the remote data source must support the ability to perform a positioned UPDATE or DELETE (WHERE CURRENT OF <cursor-name>). Some data sources do not support this capability.

> **i** Note
>
> Temporary tables cannot be updated. An UPDATE or DELETE cannot be performed if an intermediate temporary table is required. This occurs in queries with ORDER BY and some queries with subqueries.

14.3 Remote Data Access Troubleshooting

This section provides some suggestions for troubleshooting access to remote servers.

In this section:

Troubleshooting: Features not supported for remote data [page 350]
Several features are not supported for remote data.

Troubleshooting: Case sensitivity and remote data access [page 350]
The case sensitivity setting of your SAP IQ database should match the settings used by any remote servers accessed.

Troubleshooting: Connectivity tests for remote data access [page 351]
There are a few steps you can take to ensure that you can connect to a remote server.

Troubleshooting: Remote data access connections via ODBC [page 351]
If you access remote databases via ODBC, the connection to the remote server is given a name.

Remote data access on multiplex servers [page 351]
When you access a new proxy server on a secondary server, a timing problem may cause emergency server shutdown.

14.3.1 Troubleshooting: Features not supported for remote data

Several features are not supported for remote data.

- ALTER TABLE statement on remote tables.
- Triggers defined on proxy tables.
- Foreign keys that refer to remote tables.
- READTEXT, WRITETEXT, and TEXTPTR functions.
- Positioned UPDATE and DELETE statements.
- UPDATE and DELETE statements requiring an intermediate temporary table.
- Backward scrolling on cursors opened against remote data. Fetch statements must be NEXT or RELATIVE 1.
- Calls to functions that contain an expression that references a proxy table.
- If a column on a remote table has a name that is a keyword on the remote server, you cannot access data in that column. You can execute a CREATE EXISTING TABLE statement, and import the definition but you cannot select that column.
- When using remote data access, if you use an ODBC driver that does not support Unicode, then character set conversion is not performed on data coming from that ODBC driver.

14.3.2 Troubleshooting: Case sensitivity and remote data access

The case sensitivity setting of your SAP IQ database should match the settings used by any remote servers accessed.

SAP IQ databases are created case insensitive by default. With this configuration, unpredictable results may occur when selecting from a case-sensitive database. Different results will occur depending on whether ORDER BY or string comparisons are pushed off to a remote server, or evaluated by the local SAP IQ server.

14.3.3 Troubleshooting: Connectivity tests for remote data access

There are a few steps you can take to ensure that you can connect to a remote server.

- Make sure that you can connect to a remote server using a client tool such as Interactive SQL before configuring SAP IQ.
- Perform a simple passthrough statement to a remote server to check your connectivity and remote login configuration. For example:

  ```
  FORWARD TO RemoteSA {SELECT @@version};
  ```

- Turn on remote tracing for a trace of the interactions with remote servers. For example:

  ```
  SET OPTION cis_option = 7;
  ```

  Once you have turned on remote tracing, the tracing information appears in the database server messages window. You can log this output to a file by specifying the -o server option when you start the database server.

14.3.4 Troubleshooting: Remote data access connections via ODBC

If you access remote databases via ODBC, the connection to the remote server is given a name.

You can use the DROP REMOTE CONNECTION statement to cancel a remote request.

The connections are named ASACIS_<unique-database-identifier>_<conn-name>, where <conn-name> is the connection ID of the local connection. The connection ID can be obtained from the sa_conn_info stored procedure.

14.3.5 Remote data access on multiplex servers

When you access a new proxy server on a secondary server, a timing problem may cause emergency server shutdown.

If emergency server shutdown occurs, reconnect to the server, or wait for some time and start a new transaction before trying to use the new proxy table.

15 OLAP

OLAP (online analytical processing) is an efficient method of data analysis of information stored in a relational database.

Using OLAP you can analyze data on different dimensions, acquire result sets with subtotaled rows, and organize data into multidimensional cubes, all in a single SQL query. You can also use filters to drill down into the data, returning result sets quickly. This chapter describes the SQL/OLAP functionality that SAP IQ supports.

> **i** Note
>
> The tables shown in OLAP examples are available in the `iqdemo` database.

In this section:

About OLAP [page 352]
The analytic functions, which offer the ability to perform complex data analysis within a single SQL statement, are facilitated by a category of software technology named online analytical processing (OLAP).

GROUP BY Clause Extensions [page 354]
Extensions to the `GROUP BY` clause let application developers write complex SQL statements.

Analytical Functions [page 367]
SAP IQ offers both simple and windowed aggregation functions that offer the ability to perform complex data analysis within a single SQL statement.

OLAP Rules and Restrictions [page 395]
SAP IQ provides SQL OLAP functions with rules, restrictions and limitations.

Additional OLAP Examples [page 396]
This section provides additional examples using the OLAP functions.

BNF Grammar for OLAP Functions [page 405]
The Backus-Naur Form grammar outlines the specific syntactic support for the various ANSI SQL analytic functions, many of which are implemented in SAP IQ.

15.1 About OLAP

The analytic functions, which offer the ability to perform complex data analysis within a single SQL statement, are facilitated by a category of software technology named online analytical processing (OLAP).

OLAP functions are:

- `GROUP BY` clause extensions – `CUBE` and `ROLLUP`
- Analytical functions:

- Simple aggregates – `AVG`, `COUNT`, `MAX`, `MIN`, and `SUM`, `STDDEV` and `VARIANCE`

  > **i Note**
  >
  > You can use simple aggregate functions, except `Grouping()`, with an OLAP windowed function.

- Window functions:
  - Windowing aggregates – `AVG`, `COUNT`, `MAX`, `MIN`, and `SUM`
  - Ranking functions – `RANK`, `DENSE_RANK`, `PERCENT_RANK`, and `NTILE`
  - Statistical functions – `STDDEV`, `STDDEV_SAMP`, `STDDEV_POP`, `VARIANCE`, `VAR_POP`, `VAR_SAMP`, `REGR_AVGX`, `REGR_AVGY`, `REGR_COUNT`, `REGR_INTERCEPT`, `REGR_R2`, `REGR_SLOPE`, `REGR_SXX`, `REGR_SXY`, `REGR_SYY`, `CORR`, `COVAR_POP`, `COVAR_SAMP`, `CUME_DIST`, `EXP_WEIGHTED_AVG`, and `WEIGHTED_AVG`.
  - Distribution functions – `PERCENTILE_CONT` and `PERCENTILE_DISC`
- Numeric functions – `WIDTH_BUCKET`, `CEIL`, and `LN`, `EXP`, `POWER`, `SQRT`, and `FLOOR`

Extensions to the ANSI SQL standard to include complex data analysis were introduced as an amendment to the 1999 SQL standard. SAP IQ SQL enhancements support these extensions.

Some database products provide a separate OLAP module that requires you to move data from the database into the OLAP module before analyzing it. By contrast, SAP IQ builds OLAP features into the database itself, making deployment and integration with other database features, such as stored procedures, easy and seamless.

In this section:

OLAP Benefits [page 353]
OLAP functions, when combined with the `GROUPING`, `CUBE`, and `ROLLUP` extensions, provide two primary benefits.

OLAP Evaluation [page 354]
OLAP evaluation can be conceptualized as several phases of query execution that contribute to the final result.

15.1.1 OLAP Benefits

OLAP functions, when combined with the `GROUPING`, `CUBE`, and `ROLLUP` extensions, provide two primary benefits.

First, they let you perform multidimensional data analysis, data mining, time series analyses, trend analysis, cost allocations, goal seeking, ad hoc multidimensional structural changes, nonprocedural modeling, and exception alerting, often with a single SQL statement. Second, the window and reporting aggregate functions use a relational operator, called a window that can be executed more efficiently than semantically equivalent queries that use self-joins or correlated subqueries. The result sets you obtain using OLAP can have subtotal rows and can be organized into multidimensional cubes.

Moving averages and moving sums can be calculated over various intervals; aggregations and ranks can be reset as selected column values change; and complex ratios can be expressed in simple terms. Within the scope of a single query expression, you can define several different OLAP functions, each with its own partitioning rules.

15.1.2 OLAP Evaluation

OLAP evaluation can be conceptualized as several phases of query execution that contribute to the final result.

You can identify OLAP phases of execution by the relevant clause in the query. For example, if a SQL query specification contains window functions, the `WHERE`, `JOIN`, `GROUP BY`, and `HAVING` clauses are processed first. Partitions are created after the groups defined in the `GROUP BY` clause and before the evaluation of the final `SELECT` list in the query's `ORDER BY` clause.

For the purpose of grouping, all NULL values are considered to be in the same group, even though NULL values are not equal to one another.

The `HAVING` clause acts as a filter, much like the `WHERE` clause, on the results of the `GROUP BY` clause.

Consider the semantics of a simple query specification involving the SQL statements and clauses, `SELECT`, `FROM`, `WHERE`, `GROUP BY`, and `HAVING` from the ANSI SQL standard:

1. The query produces a set of rows that satisfy the table expressions present in the `FROM` clause.
2. Predicates from the `WHERE` clause are applied to rows from the table. Rows that fail to satisfy the `WHERE` clause conditions (do not equal true) are rejected.
3. Except for aggregate functions, expressions from the `SELECT` list and in the list and `GROUP BY` clause are evaluated for every remaining row.
4. The resulting rows are grouped together based on distinct values of the expressions in the `GROUP BY` clause, treating NULL as a special value in each domain. The expressions in the `GROUP BY` clause serve as partition keys if a `PARTITION BY` clause is present.
5. For each partition, the aggregate functions present in the `SELECT` list or `HAVING` clause are evaluated. Once aggregated, individual table rows are no longer present in the intermediate result set. The new result set consists of the `GROUP BY` expressions and the values of the aggregate functions computed for each partition.
6. Conditions from the `HAVING` clause are applied to result groups. Groups are eliminated that do not satisfy the `HAVING` clause.
7. Results are partitioned on boundaries defined in the `PARTITION BY` clause. OLAP windows functions (rank and aggregates) are computed for result windows.

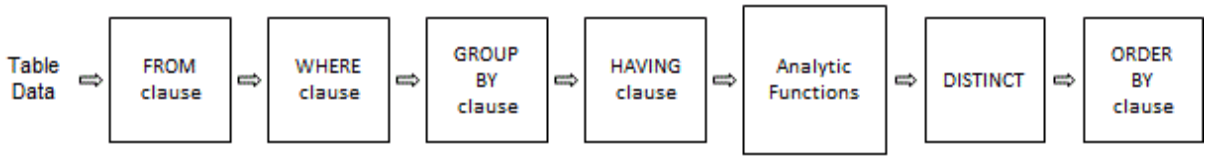


Figure 1: SQL processing for OLAP

15.2 GROUP BY Clause Extensions

Extensions to the `GROUP BY` clause let application developers write complex SQL statements.

Statements can:

- Partition the input rows in multiple dimensions and combine multiple subsets of result groups.
- Create a "data cube," providing a sparse, multi-dimensional result set for data mining analyses.
- Create a result set that includes the original groups, and optionally includes a subtotal and grand-total row.

OLAP Grouping() operations, such as `ROLLUP` and `CUBE`, can be conceptualized as prefixes and subtotal rows.

Prefixes

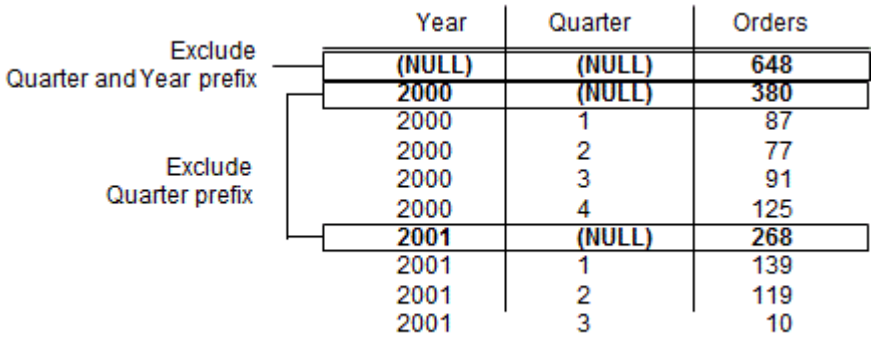
A list of prefixes is constructed for any query that contains a `GROUP BY` clause. A prefix is a subset of the items in the `GROUP BY` clause and is constructed by excluding one or more of the rightmost items from those in the query's `GROUP BY` clause. The remaining columns are called the prefix columns.

In the following `ROLLUP` example query, the `GROUP BY` list includes two variables, `<Year>` and `<Quarter>`:

```
SELECT year (OrderDate) AS Year, quarter(OrderDate)
  AS Quarter, COUNT(*) Orders
FROM SalesOrders
GROUP BY ROLLUP(Year, Quarter)
ORDER BY Year, Quarter
```

The query's two prefixes are:

- Exclude Quarter – the set of prefix columns contains the single column `Year`.
- Exclude both Quarter and Year – there are no prefix columns.



> **Note**
>
> The `GROUP BY` list contains the same number of prefixes as items.

In this section:

Group by ROLLUP and CUBE [page 355]
`ROLLUP` and `CUBE` are syntactic shortcuts that specify common grouping prefixes.

15.2.1 Group by ROLLUP and CUBE

`ROLLUP` and `CUBE` are syntactic shortcuts that specify common grouping prefixes.

In this section:

Group by ROLLUP [page 356]
The ROLLUP operator requires an ordered list of grouping expressions to be supplied as arguments.

Group by CUBE [page 364]
The CUBE operator in the GROUP BY clause analyzes data by forming the data into groups in more than one dimension (grouping expression).

15.2.1.1 Group by ROLLUP

The ROLLUP operator requires an ordered list of grouping expressions to be supplied as arguments.

ROLLUP syntax:

```
SELECT … [ GROUPING (<column-name>) … ] …
GROUP BY [ <expression> [, … ]
| ROLLUP ( <expression> [, … ] ) ]
```

GROUPING takes a column name as a parameter and returns a Boolean value as listed in the following table:

If the Value of the Result is	GROUPING Returns
NULL created by a ROLLUP operation	1 (TRUE)
NULL indicating the row is a subtotal	1 (TRUE)
Not created by a ROLLUP operation	0 (FALSE)
A stored NULL	0 (FALSE)

ROLLUP first calculates the standard aggregate values specified in the GROUP BY clause. Then ROLLUP moves from right to left through the list of grouping columns and creates progressively higher-level subtotals. A grand total is created at the end. If <n> is the number of grouping columns, then ROLLUP creates <n>+1 levels of subtotals.

The syntax GROUP BY ROLLUP (A, B, C); defines the following sets:

- (A, B, C)
- (A, B)
- (A)
- ( )

In this section:

ROLLUP and Subtotal Rows [page 357]
ROLLUP is equivalent to a UNION of a set of GROUP BY queries.

NULL Values and Subtotal Rows [page 357]
When rows in the input to a GROUP BY operation contain NULL, there is the possibility of confusion between subtotal rows added by the ROLLUP or CUBE operations and rows that contain NULL values that are part of the original input data.

15.2.1.1.1 ROLLUP and Subtotal Rows

ROLLUP is equivalent to a UNION of a set of GROUP BY queries.

The result sets of the following queries are identical. The result set of GROUP BY (A, B) consists of subtotals over all those rows in which A and B are held constant. To make a union possible, column C is assigned NULL.

This ROLLUP Query...	Is Equivalent to This Query Without ROLLUP...
`select year(orderdate) as year, quarter(orderdate) as Quarter, count(*) Ordersfrom SalesOrdersgroup by Rollup (year, quarter)order by year, quarter`	`Select null,null, count(*) Orders from SalesOrdersunion allSELECT year(orderdate) AS YEAR, NULL, count(*) Orders from SalesOrdersGROUP BY year(orderdate) union allSELECT year(orderdate) as YEAR, quarter(orderdate) as QUATER, count(*) Orders from SalesOrdersGROUP BY year(orderdate), quarter(orderdate)`

Subtotal rows can help you analyze data, especially if there are large amounts of data, different dimensions to the data, data contained in different tables, or even different databases altogether. For example, a sales manager might find reports on sales figures broken down by sales representative, region, and quarter to be useful in understanding patterns in sales. Subtotals for the data give the sales manager a picture of overall sales from different perspectives. Analyzing this data is easier when summary information is provided based on the criteria that the sales manager wants to compare.

With OLAP, the procedure for analyzing and computing row and column subtotals is invisible to users.

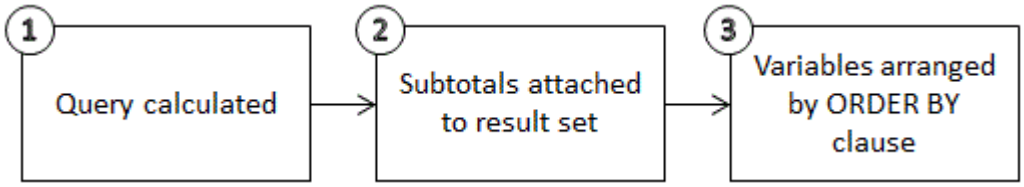


Figure 2: Subtotals

1. This step yields an intermediate result set that has not yet considered the ROLLUP.
2. Subtotals are evaluated and attached to the result set.
3. The rows are arranged according to the ORDER BY clause in the query.

15.2.1.1.2 NULL Values and Subtotal Rows

When rows in the input to a GROUP BY operation contain NULL, there is the possibility of confusion between subtotal rows added by the ROLLUP or CUBE operations and rows that contain NULL values that are part of the original input data.

The Grouping() function distinguishes subtotal rows from others by taking a column in the GROUP BY list as its argument, and returning 1 if the column is NULL because the row is a subtotal row, and 0 otherwise.

> **i Note**
>
> Examples use the SAP IQ demo database iqdemo.db.

Example 1

The following example includes Grouping() columns in the result set. Rows are highlighted that contain NULL as a result of the input data, not because they are subtotal rows. The Grouping() columns are highlighted. The query is an outer join between the `Employees` table and the `SalesOrders` table. The query selects female employees who live in Texas, New York, or California. NULL appears in the columns corresponding to those female employees who are not sales representatives (and therefore have no sales).

```
SELECT Employees.EmployeeID as EMP, year(OrderDate) as
  YEAR, count(*) as ORDERS, grouping(EMP) as
  GE, grouping(YEAR) as GY
  FROM Employees LEFT OUTER JOIN SalesOrders on
  Employees.EmployeeID = SalesOrders.SalesRepresentative
  WHERE Employees.Sex IN ('F') AND Employees.State
  IN ('TX', 'CA', 'NY')
GROUP BY ROLLUP (YEAR, EMP)
ORDER BY YEAR, EMP
```

The preceding query returns:

```
EMP         YEAR      ORDERS     GE     GY
------      ----      ------     --     --
NULL        NULL           5      1      0
NULL        NULL         169      1      1
 102        NULL           1      0      0
 309        NULL           1      0      0
1062        NULL           1      0      0
1090        NULL           1      0      0
1507        NULL           1      0      0
NULL        2000          98      1      0
 667        2000          34      0      0
 949        2000          31      0      0
1142        2000          33      0      0
NULL        2001          66      1      0
 667        2001          20      0      0
949         2001          22      0      0
1142        2001          24      0      0
```

For each prefix, a subtotal row is constructed that corresponds to all rows in which the prefix columns have the same value.

To demonstrate `ROLLUP` results, examine the example query again:

```
SELECT year (OrderDate) AS Year, quarter
  (OrderDate) AS Quarter, COUNT (*) Orders
FROM SalesOrders
  GROUP BY ROLLUP (Year, Quarter)
  ORDER BY Year, Quarter
```

In this query, the prefix containing the `Year` column leads to a summary row for Year=2000 and a summary row for Year=2001. A single summary row for the prefix has no columns, which is a subtotal over all rows in the intermediate result set.

The value of each column in a subtotal row is as follows:

- Column included in the prefix – the value of the column. For example, in the preceding query, the value of the `Year` column for the subtotal over rows with Year=2000 is 2000.
- Column excluded from the prefix – NULL. For example, the `Quarter` column has a value of NULL for the subtotal rows generated by the prefix consisting of the Year column.
- Aggregate function – an aggregate over the values of the excluded columns.

Subtotal values are computed over the rows in the underlying data, not over the aggregated rows. In many cases, such as SUM or COUNT, the result is the same, but the distinction is important in the case of statistical functions such as AVG, STDDEV, and VARIANCE, for which the result differs.

Restrictions on the ROLLUP operator are:

- The ROLLUP operator supports all of the aggregate functions available to the GROUP BY clause except COUNT DISTINCT and SUM DISTINCT.
- ROLLUP can only be used in the SELECT statement; you cannot use ROLLUP in a subquery.
- A grouping specification that combines multiple ROLLUP, CUBE, and GROUP BY columns in the same GROUP BY clause is not currently supported.
- Constant expressions as GROUP BY keys are not supported.

Example 2

The following example illustrates the use of ROLLUP and GROUPING and displays a set of mask columns created by GROUPING. The digits 0 and 1 displayed in columns S, N, and C are the values returned by GROUPING to represent the value of the ROLLUP result. A program can analyze the results of this query by using a mask of "011" to identify subtotal rows and "111" to identify the row of overall totals.

```
SELECT size, name, color, SUM(quantity),
  GROUPING(size) AS S,
  GROUPING(name) AS N,
  GROUPING(color) AS C
FROM Products
GROUP BY ROLLUP(size, name, color) HAVING (S=1 or N=1 or C=1)
ORDER BY size, name, color;
```

The preceding query returns:

```
size               name           color     SUM   S   N   C
----               -----          ------     ---   -   -   -
(NULL)             (NULL)          (NULL)    496   1   1   1
Large              (NULL)          (NULL)     71   0   1   1
Large              Sweatshirt      (NULL)     71   0   0   1
Medium             (NULL)          (NULL)    134   0   1   1
Medium             Shorts          (NULL)     80   0   0   1
Medium             Tee Shirt       (NULL)     54   0   0   1
One size fits all  (NULL)          (NULL)    263   0   1   1
One size fits all  Baseball Cap    (NULL)    124   0   0   1
One size fits all  Tee Shirt       (NULL)     75   0   0   1
One size fits all  Visor           (NULL)     64   0   0   1
Small              (NULL)          (NULL)     28   0   1   1
Small              Tee Shirt       (NULL)     28   0   0   1
```

> **i Note**
>
> In the Rollup Example 2 results, the SUM column displays as SUM(products.quantity).

Example 3

The following example illustrates the use of GROUPING to distinguish stored NULL values and "NULL" values created by the ROLLUP operation. Stored NULL values are then displayed as [NULL] in column prod_id, and "NULL" values created by ROLLUP are replaced with ALL in column PROD_IDS, as specified in the query.

```
SELECT year(ShipDate) AS Year,
               ProductID, SUM(quantity)AS OSum,
CASE
```

```
        WHEN GROUPING(Year) = 1
        THEN 'ALL'
        ELSE
        CAST(Year AS char(8))
END,
CASE
        WHEN GROUPING(ProductID) = 1
        THEN 'ALL'
        ELSE
        CAST(ProductID as char(8))
END
FROM SalesOrderItems
GROUP BY ROLLUP(Year, ProductID) HAVING OSum > 36
ORDER BY Year, ProductID;
```

The preceding query returns:

```
Year  ProductID  OSum  ...(Year)...  ...(ProductID)...
---------     -------   ---   ----------   --------
NULL           NULL   28359   ALL          ALL
2000           NULL   17642   2000         ALL
2000           300     1476   2000         300
2000           301     1440   2000         301
2000           302     1152   2000         302
2000           400     1946   2000         400
2000           401     1596   2000         401
2000           500     1704   2000         500
2000           501     1572   2000         501
2000           600     2124   2000         600
2000           601     1932   2000         601
2000           700     2700   2000         700
2001           NULL   10717   2001         ALL
2001           300      888   2001         300
2001           301      948   2001         301
2001           302      996   2001         302
2001           400     1332   2001         400
2001           401     1105   2001         401
2001           500      948   2001         500
2001           501      936   2001         501
2001           600      936   2001         600
2001           601      792   2001         601
2001           700     1836   2001         700
```

Example 4

ROLLUP example 4—The next example query returns data that summarizes the number of sales orders by year and quarter.

```
SELECT year (OrderDate) AS Year,
quarter(OrderDate) AS Quarter, COUNT (*) Orders
FROM SalesOrders
GROUP BY ROLLUP (Year, Quarter)
ORDER BY Year, Quarter
```

The following figure illustrates the query results with subtotal rows highlighted in the result set. Each subtotal row contains a NULL value in the column or columns over which the subtotal is computed.

	Year	Quarter	Orders
1	**(NULL)**	**(NULL)**	**648**
2	**2000**	**(NULL)**	**380**
3	2000	1	87
	2000	2	77
	2000	3	91
	2000	4	125
2	**2001**	**(NULL)**	**268**
3	2001	1	139
	2001	2	119
	2001	3	10

Row [1] represents the total number of orders across both years (2000, 2001) and all quarters. This row contains NULL in both the `Year` and `Quarter` columns and is the row where all columns were excluded from the prefix.

> **i** Note
>
> Every `ROLLUP` operation returns a result set with one row where NULL appears in each column except for the aggregate column. This row represents the summary of each column to the aggregate function. For example, if SUM were the aggregate function in question, this row would represent the grand total of all values.

Row [2] represent the total number of orders in the years 2000 and 2001, respectively. Both rows contain NULL in the `Quarter` column because the values in that column are rolled up to give a subtotal for `Year`. The number of rows like this in your result set depends on the number of variables that appear in your `ROLLUP` query.

The remaining rows marked [3] provide summary information by giving the total number of orders for each quarter in both years.

Example 5

This example of the `ROLLUP` operation returns a slightly more complicated result set, which summarizes the number of sales orders by year, quarter, and region. In this example, only the first and second quarters and two selected regions (Canada and the Eastern region) are examined.

```
SELECT year(OrderDate) AS Year, quarter(OrderDate)AS Quarter, region, COUNT(*)
AS Orders
FROM SalesOrders WHERE region IN ('Canada','Eastern') AND quarter IN (1, 2)
GROUP BY ROLLUP (Year, Quarter, Region)ORDER BY Year, Quarter, Region
```

The following figure illustrates the result set from the above query. Each subtotal row contains a NULL in the column or columns over which the subtotal is computed.

	Year	Quarter	Region	Orders
1	**(NULL)**	**(NULL)**	**(NULL)**	**183**
	2000	**(NULL)**	**(NULL)**	**68**
	2000	1	(NULL)	36
	2000	1	Canada	3
2	2000	1	Eastern	33
	2000	2	(NULL)	32
	2000	2	Canada	3
	2000	2	Eastern	29
	2001	**(NULL)**	**(NULL)**	**115**
	2001	1	(NULL)	57
	2001	1	Canada	11
	2001	1	Eastern	46
	2001	2	(NULL)	58
	2001	2	Canada	4
	2001	2	Eastern	54

Row [1] is an aggregate over all rows and contains NULL in the `Year`, `Quarter`, and `Region` columns. The value in the `Orders` column of this row represents the total number of orders in Canada and the Eastern region in quarters 1 and 2 in the years 2000 and 2001.

The rows marked [2] represent the total number of sales orders in each year (2000) and (2001) in quarters 1 and 2 in Canada and the Eastern region. The values of these rows [2] are equal to the grand total represented in row [1].

The rows marked [3] provide data about the total number of orders for the given year and quarter by region.

Year	Quarter	Region	Orders
(NULL)	(NULL)	(NULL)	183
2000	(NULL)	(NULL)	68
2000	**1**	**(NULL)**	**36**
2000	1	Canada	3
2000	1	Eastern	33
2000	**2**	**(NULL)**	**32**
2000	2	Canada	3
2000	2	Eastern	29
2001	(NULL)	(NULL)	**115**
2001	**1**	**(NULL)**	57
2001	1	Canada	11
2001	1	Eastern	46
2001	**2**	**(NULL)**	**58**
2001	2	Canada	4
2001	2	Eastern	54

3

The rows marked [4] provide data about the total number of orders for each year, each quarter, and each region in the result set.

Year	Quarter	Region	Orders
(NULL)	(NULL)	(NULL)	183
2000	(NULL)	(NULL)	68
2000	1	(NULL)	36
2000	**1**	**Canada**	**3**
2000	**1**	**Eastern**	**33**
2000	2	(NULL)	32
2000	**2**	**Canada**	**3**
2000	**2**	**Eastern**	**29**
2001	(NULL)	(NULL)	115
2001	1	(NULL)	57
2001	**1**	**Canada**	**11**
2001	**1**	**Eastern**	**46**
2001	2	(NULL)	58
2001	**2**	**Canada**	**4**
2001	**2**	**Eastern**	**54**

4

15.2.1.2 Group by CUBE

The CUBE operator in the GROUP BY clause analyzes data by forming the data into groups in more than one dimension (grouping expression).

CUBE requires an ordered list of dimensions as arguments and enables the SELECT statement to calculate subtotals for all possible combinations of the group of dimensions that you specify in the query and generates a result set that shows aggregates for all combinations of values in selected columns.

CUBE syntax:

```
SELECT … [ GROUPING (<column-name>) … ] …
GROUP BY [ <expression> [,…]
| CUBE ( <expression> [,…] ) ]
```

GROUPING takes a column name as a parameter, and returns a Boolean value as listed in the following table:

If the Value of the Result is	GROUPING Returns
NULL created by a CUBE operation	1 (TRUE)
NULL indicating the row is a subtotal	1 (TRUE)
Not created by a CUBE operation	0 (FALSE)
A stored NULL	0 (FALSE)

CUBE is particularly useful when your dimensions are not a part of the same hierarchy.

The SQL Syntax GROUP BY CUBE (A, B, C); defines the following sets:

- (A, B, C)
- (A, B)
- (A, C)
- (A)
- (B, C)
- (B)
- (C)
- ( )

Functional Restrictions

Restrictions on the CUBE operator are:

- The CUBE operator supports all of the aggregate functions available to the GROUP BY clause, but CUBE is currently not supported with COUNT DISTINCT or SUM DISTINCT.
- CUBE is currently not supported with the inverse distribution analytical functions, PERCENTILE_CONT and PERCENTILE_DISC.

- CUBE can only be used in the SELECT statement; you cannot use CUBE in a SELECT subquery.
- A GROUPING specification that combines ROLLUP, CUBE, and GROUP BY columns in the same GROUP BY clause is not currently supported.
- Constant expressions as GROUP BY keys are not supported.

> **i Note**
>
> CUBE performance diminishes if the size of the cube exceeds the size of the temp cache.

GROUPING can be used with the CUBE operator to distinguish between stored NULL values and NULL values in query results created by CUBE.

See the examples in the description of the ROLLUP operator for illustrations of the use of the GROUPING function to interpret results.

All CUBE operations return result sets with at least one row where NULL appears in each column except for the aggregate columns. This row represents the summary of each column to the aggregate function.

Examples

The following queries use data from a census, including the state (geographic location), gender, education level, and income of people.

Example 1

The first query contains a GROUP BY clause that organizes the results of the query into groups of rows, according to the values of the columns state, gender, and education in the table census and computes the average income and the total counts of each group. This query uses only the GROUP BY clause without the CUBE operator to group the rows.

```
SELECT State, Sex as gender, DepartmentID, COUNT(*),
  CAST(ROUND(AVG(Salary),2) AS NUMERIC(18,2)) AS AVERAGE
  FROM employees
  WHERE state IN ('MA' , 'CA')
  GROUP BY State, Sex, DepartmentID
  ORDER BY 1,2;
```

The results from the above query:

```
state  gender  DepartmentID  COUNT()     AVERAGE
-----  ------  ------------  -------     --------
CA       F              200        2     58650.00
CA       M              200        1     39300.00
```

Use the CUBE extension of the GROUP BY clause, if you want to compute the average income in the entire census of state, gender, and education and compute the average income in all possible combinations of the columns state, gender, and education, while making only a single pass through the census data. For example, use the CUBE operator if you want to compute the average income of all females in all states, or compute the average income of all people in the census according to their education and geographic location.

When CUBE calculates a group, a NULL value is generated for the columns whose group is calculated. The GROUPING function must be used to distinguish whether a NULL is a NULL stored in the database or a NULL

resulting from CUBE. The GROUPING function returns 1 if the designated column has been merged to a higher level group.

Example 2

The following query illustrates the use of the GROUPING function with GROUP BY CUBE:

```
SELECT case grouping(State) WHEN 1 THEN 'ALL'
   ELSE State
   END AS c_state, case grouping(sex)
   WHEN 1 THEN 'ALL'
   ELSE Sex end AS c_gender, case grouping(DepartmentID)
   WHEN 1 THEN 'ALL' ELSE cast(DepartmentID as char(4)) end
   AS c_dept, COUNT(*), CAST(ROUND(AVG(salary),2) AS
   NUMERIC(18,2))AS AVERAGE
   FROM employees
   WHERE state IN ('MA' , 'CA')
   GROUP BY CUBE(state, sex, DepartmentID)
   ORDER BY 1,2,3;
```

The results of this query are shown below. The NULLs generated by CUBE to indicate a subtotal row are replaced with ALL in the subtotal rows, as specified in the query:

```
c_state   c_gender  c_dept    COUNT()   AVERAGE
-------   --------  -------    -----     --------
ALL       ALL        200         3       52200.00
ALL       ALL        ALL         3       52200.00
ALL       F          200         2       58650.00
ALL       F          ALL         2       58650.00
ALL       M         200          1       39300.00
ALL       M         ALL          1       39300.00
CA        ALL       200          3       52200.00
CA        ALL       ALL          3       52200.00
CA        F         200          2       58650.00
CA        F         ALL          2       58650.00
CA        M         200          1       39300.00
CA        M         ALL          1       39300.00
```

Example 3

In this example, the query returns a result set that summarizes the total number of orders and then calculates subtotals for the number of orders by year and quarter.

```
SELECT year (OrderDate)
   AS Year, quarter(OrderDate)
   AS Quarter, COUNT (*) Orders
   FROM SalesOrders
   GROUP BY CUBE (Year, Quarter)
   ORDER BY Year, Quarter
```

> **Note**
>
> As the number of variables that you want to compare increases, the cost of computing the cube increases exponentially.

The figure that follows represents the result set from the query. The subtotal rows are highlighted in the result set. Each subtotal row has a NULL in the column or columns over which the subtotal is computed.

	Year	Quarter	Orders
1	(NULL)	(NULL)	648
2	(NULL)	1	226
	(NULL)	2	196
	(NULL)	3	101
	(NULL)	4	125
3	2000	(NULL)	380
	2000	1	87
	2000	2	77
	2000	3	91
	2000	4	125
3	2001	(NULL)	268
	2001	1	139
	2001	2	119
	2001	3	10

The first highlighted row [1] represents the total number of orders across both years and all quarters. The value in the `Orders` column is the sum of the values in each of the rows marked [3]. It is also the sum of the four values in the rows marked [2].

The next set of highlighted rows [2] represents the total number of orders by quarter across both years. The two rows marked by [3] represent the total number of orders across all quarters for the years 2000 and 2001, respectively.

15.3 Analytical Functions

SAP IQ offers both simple and windowed aggregation functions that offer the ability to perform complex data analysis within a single SQL statement.

You can use these functions to compute results for queries such as "What is the quarterly moving average of the Dow Jones Industrial average," or "List all employees and their cumulative salaries for each department." Moving averages and cumulative sums can be calculated over various intervals, and aggregations and ranks can be partitioned, so aggregate calculation is reset when partition values change. Within the scope of a single query expression, you can define several different OLAP functions, each with its own arbitrary partitioning rules. Analytical functions can be broken into two categories:

- Simple aggregate functions, such as `AVG`, `COUNT`, `MAX`, `MIN`, and `SUM` summarize data over a group of rows from the database. The groups are formed using the `GROUP BY` clause of the `SELECT` statement.
- Unary statistical aggregate functions that take one argument include `STDDEV`, `STDDEV_SAMP`, `STDDEV_POP`, `VARIANCE`, `VAR_SAMP`, and `VAR_POP`.

Both the simple and unary categories of aggregates summarize data over a group of rows from the database and can be used with a window specification to compute a moving window over a result set as it is processed.

> **i Note**
>
> The aggregate functions AVG, SUM, STDDEV, STDDEV_POP, STDDEV_SAMP, VAR_POP, VAR_SAMP, and VARIANCE do not support binary data types BINARY and VARBINARY.

In this section:

Simple Aggregate Functions [page 368]
: Simple aggregate functions, such as AVG, COUNT, MAX, MIN, and SUM summarize data over a group of rows from the database.

Windowing [page 368]
: A major feature of the ANSI SQL extensions for OLAP is a construct called a window. This windowing extension lets users divide result sets of a query (or a logical partition of a query) into groups of rows called partitions and determine subsets of rows to aggregate with respect to the current row.

Numeric Functions [page 393]
: OLAP numeric functions supported by SAP IQ include CEILING (CEIL is an alias), EXP (EXPONENTIAL is an alias), FLOOR, LN (LOG is an alias), SQRT, and WIDTH_BUCKET.

15.3.1 Simple Aggregate Functions

Simple aggregate functions, such as AVG, COUNT, MAX, MIN, and SUM summarize data over a group of rows from the database.

The groups are formed using the GROUP BY clause of the SELECT statement. These aggregates are allowed only in the select list and in the HAVING and ORDER BY clauses of a SELECT statement.

> **i Note**
>
> With the exception of Grouping() functions, both the simple and unary aggregates can be used in a windowing function that incorporates a <window clause> in a SQL query specification (a window) that conceptually creates a moving window over a result set as it is processed.

15.3.2 Windowing

A major feature of the ANSI SQL extensions for OLAP is a construct called a window. This windowing extension lets users divide result sets of a query (or a logical partition of a query) into groups of rows called partitions and determine subsets of rows to aggregate with respect to the current row.

You can use three classes of window functions with a window: ranking functions, the row numbering function, and window aggregate functions.

```
<WINDOWED TABLE FUNCTION TYPE> ::=
  <RANK FUNCTION TYPE> <LEFT PAREN> <RIGHT PAREN>
  | ROW_NUMBER <LEFT PAREN> <RIGHT PAREN>
  | <WINDOW AGGREGATE FUNCTION>
```

Windowing extensions specify a window function type over a window name or specification and are applied to partitioned result sets within the scope of a single query expression. A window partition is a subset of rows returned by a query, as defined by one or more columns in a special `OVER` clause:

```
olap_function() OVER (PARTITION BY col1, col2...)
```

Windowing operations let you establish information such as the ranking of each row within its partition, the distribution of values in rows within a partition, and similar operations. Windowing also lets you compute moving averages and sums on your data, enhancing the ability to evaluate your data and its impact on your operations.

Three Essential Parts of OLAP Windows

The OLAP windows comprise three essential aspects: window partitioning, window ordering, and window framing. Each has a significant impact on the specific rows of data visible in a window at any point in time. Meanwhile, the OLAP `OVER` clause differentiates OLAP functions from other analytic or reporting functions with three distinct capabilities:

- Defining window partitions (`PARTITION BY` clause).
- Ordering rows within partitions (`ORDER BY` clause).
- Defining window frames (ROWS/RANGE specification).

To specify multiple windows functions, and to avoid redundant window definitions, you can specify a name for an OLAP window specifications. In this usage, the keyword, `WINDOW`, is followed by at least one window definition, separated by commas. A window definition includes the name by which the window is known in the query and the details from the windows specification, which lets you to define window partitioning, ordering, and framing:

```
<WINDOW CLAUSE> ::= <WINDOW DEFINITION LIST>
```

```
<WINDOW DEFINITION LIST> ::=
  <WINDOW DEFINITION> [ { , <WINDOW DEFINITION>
  } . . . ]
```

```
<WINDOW DEFINITION> ::=
  <NEW WINDOW NAME> AS <WINDOW SPECIFICATION>
```

```
<WINDOW SPECIFICATION DETAILS> ::=
  [ <EXISTING WINDOW NAME> ]
  [ <WINDOW PARTITION CLAUSE> ]
  [ <WINDOW ORDER CLAUSE> ]
  [ <WINDOW FRAME CLAUSE> ]
```

For each row in a window partition, users can define a window frame, which may vary the specific range of rows used to perform any computation on the current row of the partition. The current row provides the reference point for determining the start and end points of the window frame.

Window specifications can be based on either a physical number of rows using a window specification that defines a window frame unit of ROWS or a logical interval of a numeric value, using a window specification that defines a window frame unit of RANGE.

Within OLAP windowing operations, you can use the following functional categories:

- Ranking functions
- Windowing aggregate functions
- Statistical aggregate functions
- Distribution functions

In this section:

Window Partitioning [page 371]
Window partitioning is the division of user-specified result sets (input rows) using a `PARTITION BY` clause.

Window Ordering [page 371]
Window ordering is the arrangement of results (rows) within each window partition using a window order clause, which contains one or more value expressions separated by commas.

Window Framing [page 372]
For nonranking aggregate OLAP functions, you can define a window frame with a window frame clause, which specifies the beginning and end of the window relative to the current row.

ROWS [page 376]
The window frame unit ROWS defines a window in the specified number of rows before or after the current row, which serves as the reference point that determines the start and end of a window.

RANGE [page 376]
Range-based window frames—The SQL/OLAP syntax supports another kind of window frame in which its limits are defined in terms of a value-based (or range-based) set of rows, rather than a specific sequence of rows.

Explicit and Inline Window Clauses [page 379]
SQL OLAP provides two ways of specifying a window in a query:

Ranking Functions [page 380]
Ranking functions let you compile a list of values from the data set in ranked order, as well as compose single-statement SQL queries that fulfill requests such as, "Name the top 10 products shipped this year by total sales," or "Give the top 5% of salespersons who sold orders to at least 15 different companies."

Ranking Examples [page 383]
These are some of the ranking functions examples:

Windowing Aggregate Functions [page 385]
Windowing aggregate functions let you manipulate multiple levels of aggregation in the same query.

Statistical Aggregate Functions [page 387]
The ANSI SQL/OLAP extensions provide a number of additional aggregate functions that permit statistical analysis of numeric data. This support includes functions to compute variance, standard deviation, correlation, and linear regression.

Interrow Functions [page 389]
The interrow functions, `LAG` and `LEAD`, provide access to previous or subsequent values in a data series, or to multiple rows in a table.

Distribution Functions [page 391]
The inverse distribution functions PERCENTILE_CONT and PERCENTILE_DISC take a percentile value as the function argument and operate on a group of data specified in the WITHIN GROUP clause or operate on the entire data set.

15.3.2.1 Window Partitioning

Window partitioning is the division of user-specified result sets (input rows) using a `PARTITION BY` clause.

A partition is defined by one or more value expressions separated by commas. Partitioned data is also implicitly sorted and the default sort order is ascending (ASC).

```
<WINDOW PARTITION CLAUSE> ::=
  PARTITION BY <WINDOW PARTITION EXPRESSION LIST>
```

If a window partition clause is not specified, then the input is treated as single partition.

> **i** Note
>
> The term partition as used with analytic functions, refers only to dividing the set of result rows using a `PARTITION BY` clause.

A window partition can be defined based on an arbitrary expression. Also, because window partitioning occurs after GROUPING (if a `GROUP BY` clause is specified), the result of any aggregate function, such as `SUM`, `AVG`, and `VARIANCE`, can be used in a partitioning expression. Therefore, partitions provide another opportunity to perform grouping and ordering operations in addition to the `GROUP BY` and `ORDER BY` clauses; for example, you can construct queries that compute aggregate functions over aggregate functions, such as the maximum `SUM` of a particular quantity.

You can specify a `PARTITION BY` clause, even if there is no `GROUP BY` clause.

15.3.2.2 Window Ordering

Window ordering is the arrangement of results (rows) within each window partition using a window order clause, which contains one or more value expressions separated by commas.

If a window order clause is not specified, the input rows could be processed in an arbitrary order.

```
<WINDOW ORDER CLAUSE> ::= <ORDER SPECIFICATION>
```

The OLAP window order clause is different from the `ORDER BY` clause that can be appended to a nonwindowed query expression.

The `ORDER BY` clause in an OLAP function, for example, typically defines the expressions for sorting rows within window partitions; however, you can use the `ORDER BY` clause without a `PARTITION BY` clause, in which case the sort specification ensures that the OLAP function is applied to a meaningful (and intended) ordering of the intermediate result set.

An order specification is a prerequisite for the ranking family of OLAP functions; it is the `ORDER BY` clause, not an argument to the function itself that identifies the measures for the ranking values. In the case of OLAP aggregates, the `ORDER BY` clause is not required in general, but it is a prerequisite to defining a window frame. This is because the partitioned rows must be sorted before the appropriate aggregate values can be computed for each frame.

The ORDER BY clause includes semantics for defining ascending and descending sorts, as well as rules for the treatment of NULL values. By default, OLAP functions assume an ascending order, where the lowest measured value is ranked 1.

Although this behavior is consistent with the default behavior of the ORDER BY clause that ends a SELECT statement, it is counterintuitive for most sequential calculations. OLAP calculations often require a descending order, where the highest measured value is ranked 1; this requirement must be explicitly stated in the ORDER BY clause with the DESC keyword.

> **i Note**
>
> Ranking functions require a <WINDOW ORDER CLAUSE> because they are defined only over sorted input. As with an <ORDER BY CLAUSE>in a <QUERY SPECIFICATION>, the default sort sequence is ascending.
>
> The use of a <WINDOW FRAME UNIT> of RANGE also requires the existence of a <WINDOW ORDER CLAUSE>. In the case of RANGE, the <WINDOW ORDER CLAUSE> may only consist of a single expression.

15.3.2.3 Window Framing

For nonranking aggregate OLAP functions, you can define a window frame with a window frame clause, which specifies the beginning and end of the window relative to the current row.

```
<WINDOW FRAME CLAUSE> ::=
  <WINDOW FRAME UNIT>
  <WINDOW FRAME EXTENT>
```

This OLAP function is computed with respect to the contents of a moving frame rather than the fixed contents of the whole partition. Depending on its definition, the partition has a start row and an end row, and the window frame slides from the starting point to the end of the partition.

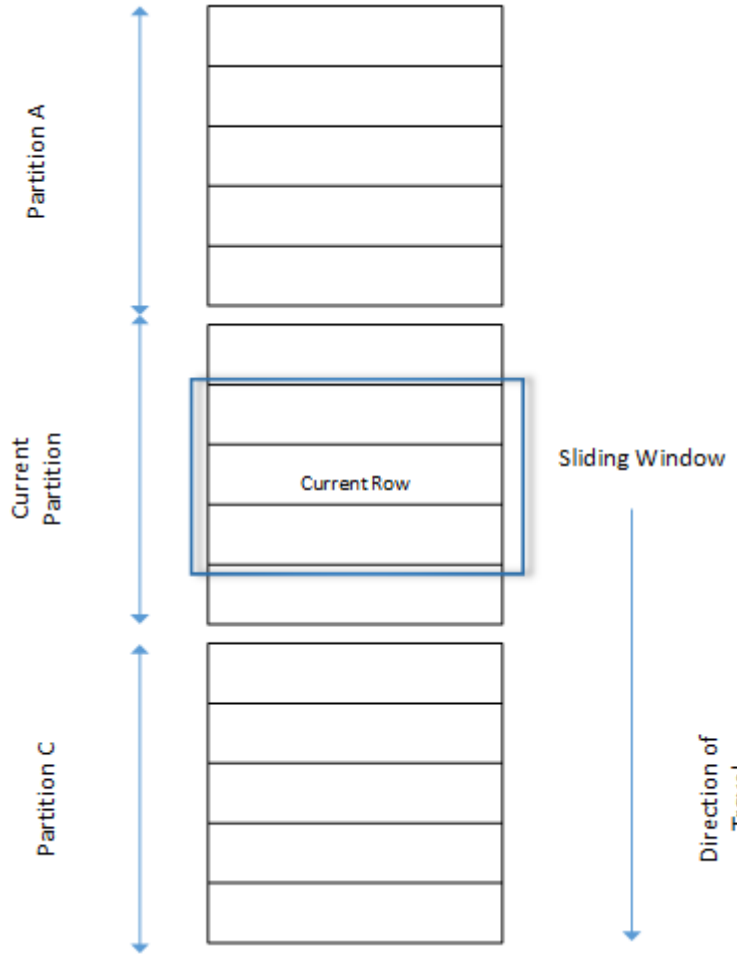


Figure 3: Three-Row Moving Window with Partitioned Input

UNBOUNDED PRECEDING and FOLLOWING

Window frames can be defined by an unbounded aggregation group that either extends back to the beginning of the partition (UNBOUNDED PRECEDING) or extends to the end of the partition (UNBOUNDED FOLLOWING), or both.

UNBOUNDED PRECEDING includes all rows within the partition preceding the current row, which can be specified with either ROWS or RANGE. UNBOUNDED FOLLOWING includes all rows within the partition following the current row, which can be specified with either ROWS or RANGE.

The value FOLLOWING specifies either the range or number of rows following the current row. If ROWS is specified, then the value is a positive integer indicating a number of rows. If RANGE is specified, the window includes any rows that are less than the current row plus the specified numeric value. For the RANGE case, the data type of the windowed value must be comparable to the type of the sort key expression of the `ORDER BY` clause. There can be only one sort key expression, and the data type of the sort key expression must allow addition.

The value PRECEDING specifies either the range or number of rows preceding the current row. If ROWS is specified, then the value is a positive integer indicating a number of rows. If RANGE is specified, the window includes any rows that are less than the current row minus the specified numeric value. For the RANGE case, the data type of the windowed value must be comparable to the type of the sort key expression of the `ORDER`

BY clause. There can be only one sort key expression, and the data type of the sort key expression must allow subtraction. This clause cannot be specified in second bound group if the first bound group is CURRENT ROW or value FOLLOWING.

The combination BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING provides an aggregate over an entire partition, without the need to construct a join to a grouped query. An aggregate over an entire partition is also known as a reporting aggregate.

CURRENT ROW Concept

In physical aggregation groups, rows are included or excluded based on their position relative to the current row, by counting adjacent rows. The current row is simply a reference to the next row in a query's intermediate results. As the current row advances, the window is reevaluated based on the new set of rows that lie within the window. There is no requirement that the current row be included in a window.

If a window frame clause is not specified, the default window frame depends on whether or not a window order clause is specified:

- If the window specification contains a window order clause, the window's start point is UNBOUNDED PRECEDING, and the end point is CURRENT ROW, thus defining a varying-size window suitable for computing cumulative values.
- If the window specification does not contain a window order clause, the window's start point is UNBOUNDED PRECEDING, and the end point is UNBOUNDED FOLLOWING, thus defining a window of fixed size, regardless of the current row.

> **i Note**
>
> A window frame clause cannot be used with a ranking function.

You can also define a window by specifying a window frame unit that is row-based (rows specification) or value-based (range specification):

```
<WINDOW FRAME UNIT> ::= ROWS | RANGE
```

```
<WINDOW FRAME EXTENT> ::= <WINDOW FRAME START> | <WINDOW FRAME BETWEEN>
```

When a window frame extent specifies BETWEEN, it explicitly provides the beginning and end of a window frame.

If the window frame extent specifies only one of these two values then the other value defaults to CURRENT ROW.

Row-based window frames—In the example rows [1] through [5] represent a partition; each row becomes the current row as the OLAP window frame slides forward. The frame is defined as Between Current Row And 2 Following, so each frame includes a maximum of three rows and a minimum of one row. When the frame reaches the end of the partition, only the current row is included. The shaded areas indicate which rows are excluded from the frame at each step.

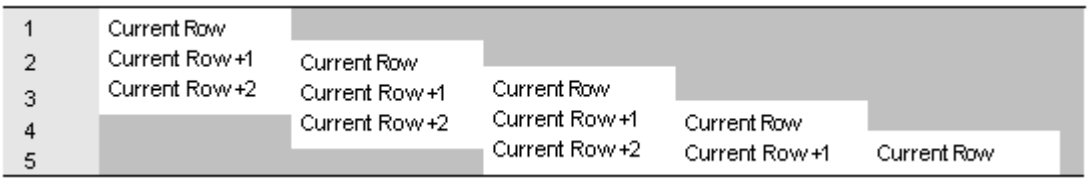
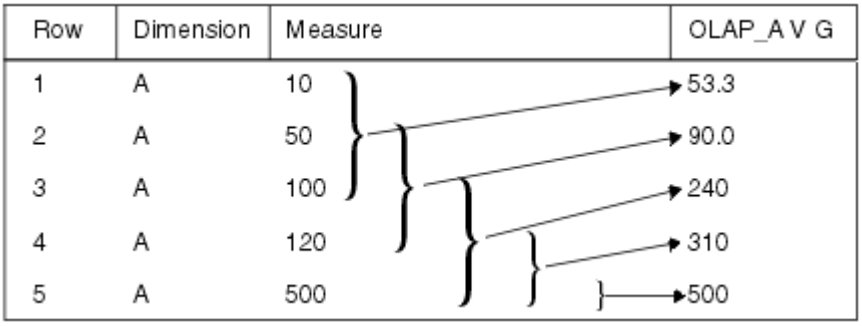


Figure 4: Row-Based Window Frames

The window frame imposes the following rules:

- When row [1] is the current row, rows [4] and [5] are excluded.
- When row [2] is the current row, rows [5] and [1] are excluded.
- When row [3] is the current row, rows [1] and [2] are excluded.
- When row [4] is the current row, rows [1], [2], and [3] are excluded.
- When row [5] is the current row, rows [1], [2], [3], and [4] are excluded.

The following diagram applies these rules to a specific set of values, showing the OLAP `AVG` function that would be calculated for each row. The sliding calculations produce a moving average with an interval of three rows or fewer, depending on which row is the current row:



The following example demonstrates a sliding window:

```
SELECT dimension, measure,
  AVG(measure) OVER(partition BY dimension
    ORDER BY measure
    ROWS BETWEEN CURRENT ROW and 2 FOLLOWING)
    AS olap_avg
FROM ...
```

The averages are computed as follows:

- Row [1] = (10 + 50 + 100)/3
- Row [2] = (50+ 100 + 120)/3
- Row [3] = (100 + 120 + 500)/3
- Row [4] = (120 + 500 + NULL)/3
- Row [5] = (500 + NULL + NULL)/3

Similar calculations would be computed for all subsequent partitions in the result set (such as, B, C, and so on).

If there are no rows in the current window, the result is NULL, except for `COUNT`.

15.3.2.4 ROWS

The window frame unit ROWS defines a window in the specified number of rows before or after the current row, which serves as the reference point that determines the start and end of a window.

Each analytical calculation is based on the current row within a partition. To produce determinative results for a window expressed in rows, the ordering expression should be unique.

The reference point for all window frames is the current row. The SQL/OLAP syntax provides mechanisms for defining a row-based window frame as any number of rows preceding or following the current row or preceding and following the current row.

The following list illustrates common examples of a window frame unit:

- Rows between unbounded preceding and current row – specifies a window whose start point is the beginning of each partition and the end point is the current row and is often used to construct windows that compute cumulative results, such as cumulative sums.
- Rows between unbounded preceding and unbounded following – specifies a fixed window, regardless of the current row, over the entire partition. The value of a window aggregate function is, therefore, identical in each row of the partition.
- Rows between 1 preceding and 1 following – specifies a fixed-sized moving window over three adjacent rows, one each before and after the current row. You can use this window frame unit to compute, for example, a 3-day or 3-month moving average.
  Be aware of meaningless results that may be generated by gaps in the windowed values when using ROWS. If the set of values is not continuous, consider using RANGE instead of ROWS, because a window definition based on RANGE automatically handles adjacent rows with duplicate values and does not include other rows when there are gaps in the range.

  > **i Note**
  >
  > In the case of a moving window, it is assumed that rows containing NULL values exist before the first row, and after the last row, in the input. This means that in a 3-row moving window, the computation for the last row in the input—the current row— includes the immediately preceding row and a NULL value.
- Rows between current row and current row – restricts the window to the current row only.
- Rows between 1 preceding and 1 preceding – specifies a single row window consisting only of the preceding row, with respect to the current row. In combination with another window function that computes a value based on the current row only, this construction makes it possible to easily compute deltas, or differences in value, between adjacent rows.

15.3.2.5 RANGE

Range-based window frames—The SQL/OLAP syntax supports another kind of window frame in which its limits are defined in terms of a value-based (or range-based) set of rows, rather than a specific sequence of rows.

Value-based window frames define rows within a window partition that contain a specific range of numeric values. The OLAP function's `ORDER BY` clause defines the numeric column to which the range specification is applied, relative to the current row's value for that column. The range specification uses the same syntax as the rows specification, but the syntax is interpreted in a different way.

The window frame unit, RANGE, defines a window frame whose contents are determined by finding rows in which the ordering column has values within the specified range of value relative to the current row. This is

called a logical offset of a window frame, which you can specify with constants, such as “3 preceding,” or any expression that can be evaluated to a numeric constant. When using a window defined with RANGE, there can be only a single numeric expression in the `ORDER BY` clause.

> **i Note**
>
> `ORDER BY` key must be a numeric data in RANGE window frame

For example, a frame can be defined as the set of rows with `<year>` values some number of years preceding or following the current row’s year:

```
ORDER BY year ASC range BETWEEN 1 PRECEDING AND CURRENT ROW
```

The phrase 1 PRECEDING means the current row’s `<year>` value minus 1.

This kind of range specification is inclusive. If the current row’s `<year>` value is 2000, all rows in the window partition with year values 2000 and 1999 qualify for the frame, regardless of the physical position of those rows in the partition. The rules for including and excluding value-based rows are quite different from the rules applied to row-based frames, which depend entirely on the physical sequence of rows.

Put in the context of an OLAP `AVG()` calculation, the following partial result set further demonstrates the concept of a value-based window frame. Again, the frame consists of rows that:

- Have the same year as the current row
- Have the same year as the current row minus 1

Row	Dimension	Year	Measure	Olap_avg
1	A	1999	10000	10000
2	A	2001	5000	3000
3	A	2001	1000	3000
4	A	2002	12000	5250
5	A	2002	3000	5250

The following query demonstrates a range-based window definition:

```
SELECT dimension, year, measure,
  AVG(measure) OVER(PARTITION BY dimension
    ORDER BY year ASC
    range BETWEEN CURRENT ROW and 1 PRECEDING)
    as olap_avg
FROM ...
```

The averages are computed as follows:

- Row [1] = 1999; rows [2] through [5] are excluded; AVG = 10,000/1
- Row [2] = 2001; rows [1], [4], and [5] are excluded; AVG = 6,000/2
- Row [3] = 2001; rows [1], [4], and [5] are excluded; AVG = 6,000/2
- Row [4] = 2002; row [1] is excluded; AVG = 21,000/4
- Row [5] = 2002; row [1] is excluded; AVG = 21,000/4

Ascending and descending order for value -based frames—The `ORDER BY` clause for an OLAP function with a value-based window frame not only identifies the numeric column on which the range specification is based; it

also declares the sort order for the `ORDER BY` values. The following specification is subject to the sort order that precedes it (ASC or DESC):

```
RANGE BETWEEN CURRENT ROW AND <n> FOLLOWING
```

The specification `<n>` FOLLOWING means:

- Plus `<n>` if the partition is sorted in default ascending order (ASC)
- Minus `<n>` if the partition is sorted in descending order (DESC)

For example, assume that the `year` column contains four distinct values, from 1999 to 2002. The following table shows the default ascending order of these values on the left and the descending order on the right:

ORDER BY year ASC	ORDER BY year DESC
1999	2002
2000	2001
2001	2000
2002	1999

If the current row is 1999 and the frame is specified as follows, rows that contain the values 1999 and 1998 (which does not exist in the table) are included in the frame:

```
ORDER BY year DESC range BETWEEN CURRENT ROW and 1 FOLLOWING
```

> **i Note**
>
> The sort order of the `ORDER BY` values is a critical part of the test for qualifying rows in a value-based frame; the numeric values alone do not determine exclusion or inclusion.

Using an unbounded window—The following query produces a result set consisting of all of the products accompanied by the total quantity of all products:

```
SELECT id, description, quantity,
  SUM(quantity) OVER () AS total
  FROM products;
```

Computing deltas between adjacent rows—Using two windows—one over the current row and the other over the previous row—provides a direct way of computing deltas, or changes, between adjacent rows.

```
SELECT EmployeeID, Surname, SUM(salary)
OVER(ORDER BY BirthDate rows between current row and current row)
AS curr, SUM(Salary)
OVER(ORDER BY BirthDate rows between 1 preceding and 1 preceding)
AS prev, (curr-prev) as delta
FROM Employees
WHERE State IN ('MA', 'AZ', 'CA', 'CO') AND DepartmentID>10
ORDER BY EmployeeID, Surname;
```

The results from the query:

```
EmployeeID  Surname      curr         prev         delta
---------   ---------    --------     ---------    -----
148         Jordan       51432.000191
209         Bertrand              29800.000            39300.000
-9500.000278
```

```
225         Melkisetian     48500.000       42300.000
6200.000299
657         Overbey                 39300.000           41700.750
-2400.750318
902         Crow                    41700.750           45000.000
-3299.250586
949         Coleman                 42300.000           46200.000
-3900.000690
1053        Poitras                 46200.000           29800.000
16400.000703
1090        Martinez                55500.800           51432.000
4068.800949
1154        Savarino                72300.000           55500.800
16799.2001101
1420        Preston     37803.000           48500.000
-10697.0001142
1507        Clark       45000.000           72300.000    -27300.000
```

Although the window function `SUM()` is used, the sum contains only the salary value of either the current or previous row because of the way the window is specified. Also, the `prev` value of the first row in the result is NULL because it has no predecessor; therefore, the `delta` is NULL as well.

In each of the examples above, the function used with the `OVER()` clause is the `SUM()` aggregate function.

15.3.2.6 Explicit and Inline Window Clauses

SQL OLAP provides two ways of specifying a window in a query:

- The explicit window clause lets you define a window that follows a `HAVING` clause. You reference windows defined with those window clauses by specifying their names when you invoke an OLAP function, such as:

  ```
  SUM ( ...) OVER w2
  ```

- The inline window specification lets you define a window in the `SELECT` list of a query expression. This capability lets you define your windows in a window clause that follows the `HAVING` clause and then reference them by name from your window function invocations, or to define them along with the function invocations.

  > **i Note**
  >
  > If you use an inline window specification, you cannot name the window. Two or more window function invocations in a single `SELECT` list that use identical windows must either reference a named window defined in a window clause or they must define their inline windows redundantly.

Example

This example shows a window function:

```
SELECT DepartmentID, Surname, StartDate, Salary, SUM(Salary) OVER
(PARTITION BY DepartmentID ORDER BY startdate
rows between unbounded preceding and current row)
AS sum_salary FROM Employees
WHERE State IN ('CA') AND DepartmentID IN (100, 200)
```

```
ORDER BY DepartmentID;
```

The query returns a result set that partitions the data by department and then provides a cumulative summary of employees' salaries, starting with the employee who has been at the company the longest. The result set includes only those employees who reside in Massachusetts. The column `sum_salary` provides the cumulative total of employees' salaries.

The following result set is partitioned by department:

```
DepartmentID  Surname   start_date  salary     sum_salary
------------  --------  ----------  ---------  ----------
200           Overbey   1987-02-19  39300.000   39300.000
200           Savarino  1989-11-07  72300.000  111600.000
200           Clark     1990-07-21  45000.000  156600.000
```

15.3.2.7 Ranking Functions

Ranking functions let you compile a list of values from the data set in ranked order, as well as compose single-statement SQL queries that fulfill requests such as, "Name the top 10 products shipped this year by total sales," or "Give the top 5% of salespersons who sold orders to at least 15 different companies."

SQL/OLAP defines five functions that are categorized as ranking functions:

```
<RANK FUNCTION TYPE> ::=
  RANK | DENSE_RANK | PERCENT_RANK | ROW_NUMBER | NTILE
```

Ranking functions let you compute a rank value for each row in a result set based on the order specified in the query. For example, a sales manager might need to identify the top or bottom sales people in the company, the highest- or lowest-performing sales region, or the best- or worst-selling products. Ranking functions can provide this information.

In this section:

RANK [page 381]
: The `RANK` function returns a number that indicates the rank of the current row among the rows in the row's partition, as defined by the `ORDER BY` clause.

DENSE_RANK [page 381]
: `DENSE_RANK` returns ranking values without gaps.

PERCENT_RANK [page 382]
: The `PERCENT_RANK` function calculates a percentage for the rank, rather than a fractional amount, and returns a decimal value between 0 and 1.

ROW_NUMBER [page 382]
: The `ROW_NUMBER` function returns a unique row number for each row.

15.3.2.7.1 RANK

The RANK function returns a number that indicates the rank of the current row among the rows in the row's partition, as defined by the ORDER BY clause.

The first row in a partition has a rank of 1, and the last rank in a partition containing 25 rows is 25. RANK is specified as a syntax transformation, which means that an implementation can choose to actually transform RANK into its equivalent, or it can merely return a result equivalent to the result that transformation would return.

In the following example, ws1 indicates the window specification that defines the window named w1:

```
RANK() OVER ws
```

The example is equivalent to the following:

```
( COUNT (*) OVER ( ws RANGE UNBOUNDED PRECEDING )
- COUNT (*) OVER ( ws RANGE CURRENT ROW ) + 1 )
```

The transformation of the RANK function uses logical aggregation (RANGE). As a result, two or more records that are tied—or have equal values in the ordering column—have the same rank. The next group in the partition that has a different value has a rank that is more than one greater than the rank of the tied rows. For example, if there are rows whose ordering column values are 10, 20, 20, 20, 30, the rank of the first row is 1 and the rank of the second row is 2. The rank of the third and fourth row is also 2, but the rank of the fifth row is 5. There are no rows whose rank is 3 or 4. This algorithm is sometimes known as sparse ranking.

See *RANK Function [Analytical]* in *SAP IQ SQL Reference* for complete syntax information.

15.3.2.7.2 DENSE_RANK

DENSE_RANK returns ranking values without gaps.

The values for rows with ties are still equal, but the ranking of the rows represents the positions of the clusters of rows having equal values in the ordering column, rather than the positions of the individual rows. As in the RANK example, where rows ordering column values are 10, 20, 20, 20, 30, the rank of the first row is still 1 and the rank of the second row is still 2, as are the ranks of the third and fourth rows. The last row, however, is 3, not 5.

DENSE_RANK is computed through a syntax transformation, as well:

```
DENSE_RANK() OVER ws
```

The statement is equivalent to the following, where <expr_1> through <expr_n> represent the list of value expressions in the sort specification list of window w1:

```
COUNT ( DISTINCT ROW ( expr_1, . . ., expr_n ) )
 OVER ( ws RANGE UNBOUNDED PRECEDING )
```

See *DENSE_RANK Function [Analytical]* in *SAP IQ SQL Reference* for complete syntax information.

15.3.2.7.3 PERCENT_RANK

The PERCENT_RANK function calculates a percentage for the rank, rather than a fractional amount, and returns a decimal value between 0 and 1.

PERCENT_RANK returns the relative rank of a row, which is a number that indicates the relative position of the current row within the window partition in which it appears. For example, in a partition that contains 10 rows having different values in the ordering columns, the third row is given a PERCENT_RANK value of 0.222 ..., because you have covered 2/9 (22.222...%) of rows following the first row of the partition. PERCENT_RANK of a row is defined as one less than the RANK of the row divided by one less than the number of rows in the partition, as seen in the following example (where "ANT" stands for an approximate numeric type, such as REAL or DOUBLE PRECISION).

```
PERCENT_RANK() OVER ws
```

The syntax is equivalent to the following:

```
CASE
  WHEN COUNT (*) OVER ( ws RANGE BETWEEN UNBOUNDED
  PRECEDING AND UNBOUNDED FOLLOWING ) = 1
  THEN CAST (0 AS ANT)
  ELSE
   ( CAST ( RANK () OVER ( ws ) AS ANT ) -1 /
   ( COUNT (*) OVER ( ws RANGE BETWEEN UNBOUNDED
   PRECEDING AND UNBOUNDED FOLLOWING ) - 1 )
END
```

See *PERCENT_RANK Function [Analytical]* in *SAP IQ SQL Reference* for complete syntax information.

15.3.2.7.4 ROW_NUMBER

The ROW_NUMBER function returns a unique row number for each row.

If you define window partitions, ROW_NUMBER starts the row numbering in each partition at 1, and increments each row by 1. If you do not specify a window partition, ROW_NUMBER numbers the complete result set from 1 to the total cardinality of the table.

The ROW_NUMBER function syntax is:

```
ROW_NUMBER( ) OVER
    ( [ PARTITION BY <window partition> ]
    ORDER BY <window ordering> )
```

ROW_NUMBER does not require an argument, but you must specify the parentheses.

The PARTITION BY clause is optional. The OVER (ORDER_BY) clause cannot contain a window frame ROWS/RANGE specification.

See *ROW_NUMBER Function [Analytical]* in *SAP IQ SQL Reference* for complete syntax information.

15.3.2.8 Ranking Examples

These are some of the ranking functions examples:

Example 1

This query finds the male and female employees from California, and ranks them in descending order according to salary:

```
SELECT Surname, Sex, Salary, RANK() OVER (
ORDER BY Salary DESC) as RANK FROM Employees
WHERE State IN ('CA') AND DepartmentID =200
ORDER BY Salary DESC;
```

The query results in the following:

```
Surname      Sex        Salary    RANK
-------      ---        ------    ----
Savarino      F          72300.000  1
Clark         F          45000.000  2
Overbey       M          39300.000  3
```

Example 2

Using the query from the previous example, you can change the data by partitioning it by gender. This example ranks employees in descending order by salary and partitions by gender:

```
SELECT Surname, Sex, Salary, RANK() OVER (PARTITION BY Sex
ORDER BY Salary DESC) AS RANK FROM Employees
WHERE State IN ('CA', 'AZ') AND DepartmentID IN (200, 300)
ORDER BY Sex, Salary DESC;
```

The query results in the following:

```
Surname           Sex      Salary     RANK
-------           ---      ---------  ----
Savarino           F        72300.000  1
Jordan             F        51432.000  2
Clark              F        45000.000  3
Coleman            M        42300.000  1
Overbey            M        39300.000  2
```

Example 3

This example ranks a list of female employees in California and Texas in descending order according to salary:

```
SELECT Surname, Salary, Sex, CAST(PERCENT_RANK() OVER
```

```
(ORDER BY Salary DESC) AS numeric (4, 2)) AS RANK
FROM Employees WHERE State IN ('CA', 'TX') AND Sex ='F'
ORDER BY Salary DESC;
```

The PERCENT_RANK function provides the cumulative total in descending order.

The query results in the following:

```
Surname         salary   sex           RANK
---------      ---------   ---   ----------
Savarino       72300.000   F           0.00
Smith          51411.000   F           0.33
Clark          45000.000   F           0.66
Garcia         39800.000   F           1.00
```

Example 4

You can use the PERCENT_RANK function to find the top or bottom percentiles in the data set. This query returns female employees whose salary is in the top five percent of the data set:

```
SELECT * FROM (SELECT Surname, Salary, Sex,
CAST(PERCENT_RANK() OVER (ORDER BY salary DESC) as
numeric (4, 2)) AS percent
FROM Employees WHERE State IN ('CA') AND sex ='F' ) AS
DT where percent > 0.5
ORDER BY Salary DESC;
```

The query results in the following:

```
Surname        salary        sex   percent
---------      ----------    ---   ---------
Clark          45000.000     F     1.00
```

Example 5

This example uses the ROW_NUMBER function to return row numbers for each row in all window partitions. The query partitions the Employees table by department ID, and orders the rows in each partition by start date:

```
SELECT DepartmentID dID, StartDate, Salary ,
ROW_NUMBER()OVER(PARTITION BY dID ORDER BY StartDate)
FROM  Employees ORDER BY 1,2;
```

The query results in the following:

```
dID        StartDate     Salary      Row_number()
========   ===========   ==========  =============
  100      1984-08-28    47500.000       1
  100      1985-01-01    62000.500       2
  100      1985-06-17    57490.000       3
  100      1986-06-07    72995.000       4
  100      1986-07-01    48023.690       5
  ...      ...           ...             ...
  200      1985-02-03    38500.000       1
```

```
  200      1985-12-06    54800.000       2
  200      1987-02-19    39300.000       3
  200      1987-07-10    49500.000       4
  ...      ...           ...             ...
  500      1994-02-27     24903.000      9
```

15.3.2.9 Windowing Aggregate Functions

Windowing aggregate functions let you manipulate multiple levels of aggregation in the same query.

For example, you can list all quarters during which expenses are less than the average. You can use aggregate functions, including the simple aggregate functions `AVG`, `COUNT`, `MAX`, `MIN`, and `SUM`, to place results — possibly computed at different levels in the statement — on the same row. This placement provides a means to compare aggregate values with detail rows within a group, avoiding the need for a join or a correlated subquery.

These functions also let you compare nonaggregate values to aggregate values. For example, a salesperson might need to compile a list of all customers who ordered more than the average number of a product in a specified year, or a manager might want to compare an employee's salary against the average salary of the department.

If a query specifies `DISTINCT` in the `SELECT` statement, then the `DISTINCT` operation is applied after the window operator. A window operator is computed after processing the `GROUP BY` clause and before the evaluation of the `SELECT` list items and a query's `ORDER BY` clause.

Examples

Example 1

This query returns a result set, partitioned by year, that shows a list of the products that sold higher-than-average sales:

```
SELECT * FROM (SELECT Surname AS E_name, DepartmentID AS
   Dept, CAST(Salary AS numeric(10,2) ) AS Sal,
   CAST(AVG(Sal) OVER(PARTITION BY DepartmentID) AS
   numeric(10, 2)) AS Average, CAST(STDDEV_POP(Sal)
   OVER(PARTITION BY DepartmentID) AS numeric(10,2)) AS
   STD_DEV
FROM Employees
GROUP BY Dept, E_name, Sal) AS derived_table WHERE
   Sal> (Average+STD_DEV )
ORDER BY Dept, Sal, E_name;
```

The query results in the following:

```
E_name    Dept  Sal       Average   STD_DEV
--------  ----  --------  -------   --------
Lull      100         87900.00        58736.28        16829.59
Sheffield 100         87900.00        58736.28        16829.59
Scott              100         96300.00        58736.28        16829.59
Sterling     200         64900.00        48390.94        13869.59
Savarino     200         72300.00        48390.94     13869.59
Kelly              200         87500.00        48390.94     13869.59
Shea               300        138948.00        59500.00        30752.39
```

```
Blaikie          400          54900.00         43640.67     11194.02
Morris              400          61300.00         43640.67     11194.02
Evans               400          68940.00         43640.67     11194.02
Martinez     500          55500.80         33752.20          9084.49
```

For the year 2000, the average number of orders was 1,787. Four products (700, 601, 600, and 400) sold higher than that amount. In 2001, the average number of orders was 1,048 and 3 products exceeded that amount.

Example 2

This query returns a result set that shows the employees whose salary is one standard deviation greater than the average salary of their department. Standard deviation is a measure of how much the data varies from the mean.

```
SELECT * FROM (SELECT Surname AS E_name, DepartmentID AS
  Dept, CAST(Salary AS numeric(10,2) ) AS Sal,
  CAST(AVG(Sal) OVER(PARTITION BY dept) AS
  numeric(10, 2)) AS Average, CAST(STDDEV_POP(Sal)
  OVER(PARTITION BY dept) AS numeric(10,2)) AS
  STD_DEV
FROM Employees
GROUP BY Dept, E_name, Sal) AS derived_table WHERE
  Sal> (Average+STD_DEV )
ORDER BY Dept, Sal, E_name;
```

Every department has at least one employee whose salary significantly deviates from the mean, as shown in these results:

```
E_name       Dept   Sal        Average      STD_DEV
--------     ----   --------   --------     --------
Lull          100   87900.00   58736.28     16829.59
Sheffield     100   87900.00   58736.28     16829.59
Scott         100   96300.00   58736.28     16829.59
Sterling      200   64900.00   48390.94     13869.59
Savarino      200   72300.00   48390.94     13869.59
Kelly         200   87500.00   48390.94     13869.59
Shea          300  138948.00   59500.00     30752.39
Blaikie       400   54900.00   43640.67     11194.02
Morris        400   61300.00   43640.67     11194.02
Evans         400   68940.00   43640.67     11194.02
Martinez      500   55500.80   33752.20      9084.49
```

Employee Scott earns $96,300.00, while the average salary for department 100 is $58,736.28. The standard deviation for department 100 is 16,829.00, which means that salaries less than $75,565.88 (58736.28 + 16829.60 = 75565.88) fall within one standard deviation of the mean.

15.3.2.10 Statistical Aggregate Functions

The ANSI SQL/OLAP extensions provide a number of additional aggregate functions that permit statistical analysis of numeric data. This support includes functions to compute variance, standard deviation, correlation, and linear regression.

Standard Deviation and Variance

The SQL/OLAP general set functions that take one argument include those appearing in this syntax statement:

```
<SIMPLE WINDOW AGGREGATE FUNCTION TYPE> ::=
  <BASIC AGGREGATE FUNCTION TYPE>
  | STDDEV | STDDEV_POP | STDDEV_SAMP
  | VARIANCE | VARIANCE_POP | VARIANCE_SAMP
```

- `STDDEV_POP` – computes the population standard deviation of the provided value expression evaluated for each row of the group or partition (if `DISTINCT` is specified, each row that remains after duplicates are eliminated), defined as the square root of the population variance.
- `STDDEV_SAMP` – computes the population standard deviation of the provided value expression evaluated for each row of the group or partition (if `DISTINCT` is specified, each row that remains after duplicates are eliminated), defined as the square root of the sample variance.
- `VAR_POP` – computes the population variance of value expression evaluated for each row of the group or partition (if `DISTINCT` is specified, each row that remains after duplicates are eliminated), defined as the sum of squares of the difference of value expression from the mean of value expression, divided by the number of rows (remaining) in the group or partition.
- `VAR_SAMP` – computes the sample variance of value expression evaluated for each row of the group or partition (if `DISTINCT` is specified, each row that remains after duplicates are eliminated), defined as the sum of squares of the difference of value expression, divided by one less than the number of rows (remaining) in the group or partition.

These functions, including `STDDEV` and `VARIANCE`, are true aggregate functions in that they can compute values for a partition of rows as determined by the query's `ORDER BY` clause. As with other basic aggregate functions such as `MAX` or `MIN`, their computation ignores NULL values in the input. Also, regardless of the domain of the expression being analyzed, all variance and standard deviation computation uses IEEE double-precision floating point. If the input to any variance or standard deviation function is the empty set, then each function returns NULL as its result. If `VAR_SAMP` is computed for a single row, it returns NULL, while `VAR_POP` returns the value 0.

Correlation

The SQL/OLAP function that computes a correlation coefficient is:

- `CORR` – returns the correlation coefficient of a set of number pairs.

You can use the CORR function either as a windowing aggregate function (where you specify a window function type over a window name or specification) or as a simple aggregate function with no OVER clause.

Covariance

The SQL/OLAP functions that compute covariances include:

- `COVAR_POP` – returns the population covariance of a set of number pairs.
- `COVAR_SAMP` – returns the sample covariance of a set of number pairs.

The covariance functions eliminate all pairs where expression1 or expression2 has a null value.

You can use the covariance functions either as windowing aggregate functions (where you specify a window function type over a window name or specification) or as simple aggregate functions with no OVER clause.

Cumulative Distribution

The SQL/OLAP function that calculates the relative position of a single value among a group of rows is `CUME_DIST`.

The window specification must contain an ORDER_BY clause.

Composite sort keys are not allowed in the CUME_DIST function.

Regression Analysis

The regression analysis functions calculate the relationship between an independent variable and a dependent variable using a linear regression equation. The SQL/OLAP linear regression functions include:

- `REGR_AVGX` – computes the average of the independent variable of the regression line.
- `REGR_AVGY` – computes the average of the dependent variable of the regression line.
- `REGR_COUNT` – returns an integer representing the number of nonnull number pairs used to fit the regression line.
- `REGR_INTERCEPT` – computes the y-intercept of the regression line that best fits the dependent and independent variables.
- `REGR_R2` – computes the coefficient of determination (the goodness-of-fir statistic) for the regression line.
- `REGR_SLOPE` – computes the slope of the linear regression line fitted to nonnull pairs.
- `REGR_SXX` – returns the sum of squares of the independent expressions used in a linear regression model. Use this function to evaluate the statistical validity of the regression model.
- `REGR_SXY` – returns the sum of products of the dependent and independent variables. Use this function to evaluate the statistical validity of the regression model.
- `REGR_SYY` – returns values that can evaluate the statistical validity of a regression model.

You can use the regression analysis functions either as windowing aggregate functions (where you specify a window function type over a window name or specification) or as simple aggregate functions with no OVER clause.

Weighted OLAP Aggregates

The weighted OLAP aggregate functions calculate weighted moving averages:

- `EXP_WEIGHTED_AVG` – calculates an exponentially weighted moving average. Weightings determine the relative importance of each quantity comprising the average. Weights in `EXP_WEIGHTED_AVG` decrease exponentially. Exponential weighting applies more weight to the most recent values and decreases the weight for older values, while still applying some weight
- `WEIGHTED_AVG` – calculates a linearly weighted moving average where weights decrease arithmetically over time. Weights decrease from the highest weight for the most recent data points, down to zero for the oldest data point.

The window specification must contain an `ORDER_BY` clause.

Nonstandard Database Industry Extensions

Non-ANSI SQL/OLAP aggregate function extensions used in the database industry include `FIRST_VALUE`, `MEDIAN`, and `LAST_VALUE`:

- `FIRST_VALUE` – returns the first value from a set of values.
- `MEDIAN` – returns the median from an expression.
- `LAST_VALUE` – returns the last value from a set of values.

The `FIRST_VALUE` and `LAST_VALUE` functions require a window specification. You can use the `MEDIAN` function either as windowing aggregate function (where you specify a window function type over a window name or specification) or as a simple aggregate function with no OVER clause.

15.3.2.11 Interrow Functions

The interrow functions, `LAG` and `LEAD`, provide access to previous or subsequent values in a data series, or to multiple rows in a table.

Interrow functions also partition simultaneously without a self-join. LAG provides access to a row at a given physical offset prior to the `CURRENT ROW` in the table or partition. LEAD provides access to a row at a given physical offset after the `CURRENT ROW` in the table or partition.

`LAG` and `LEAD` syntax is identical. Both functions require an `OVER` (`ORDER_BY`) window specification.

- `LAG` syntax:

```
LAG (<value_expr>) [, <offset> [, <default>]])
     OVER ([PARTITION BY <window partition>] ORDER BY <window ordering>)
```

- `LEAD` syntax:

```
LEAD (<value_expr>) [, <offset> [, <default>]])
    OVER ([PARTITION BY <window partition>] ORDER BY <window ordering>)
```

The PARTITION BY clause in the OVER (ORDER_BY) clause is optional. The OVER (ORDER_BY) clause cannot contain a window frame ROWS/RANGE specification.

<value_expr >is a table column or expression that defines the offset data to return from the table. You can define other functions in the <value_expr>, with the exception of analytic functions.

For both functions, specify the target row by entering a physical offset. The <offset >value is the number of rows above or below the current row. Enter a non-negative numeric data type (entering a negative value generates an error). If you enter 0, SAP IQ returns the current row.

The optional <default> value defines the value to return if the <offset >value goes beyond the scope of the table. The default value of <default> is NULL. The data type of <default> must be implicitly convertible to the data type of the <value_expr> value, or SAP IQ generates a conversion error.

The inter-row functions are useful in financial services applications that perform calculations on data streams, such as stock transactions. The following example uses the LAG function to calculate the percentage change in the trading price of a particular stock. Consider the following trading data from a fictional table called stock_trades:

```
traded at            symbol   price
-------------------  ------   ------
2009-07-13 06:07:12  SQL      15.84
2009-07-13 06:07:13  TST       5.75
2009-07-13 06:07:14  TST       5.80
2009-07-13 06:07:15  SQL      15.86
2009-07-13 06:07:16  TST       5.90
2009-07-13 06:07:17  SQL      15.86
```

> **Note**
>
> The fictional stock_trades table is not available in the iqdemo database.

The query partitions the trades by stock symbol, orders them by time of trade, and uses the LAG function to calculate the percentage increase or decrease in trade price between the current trade and the previous trade:

```
select  stock_symbol as 'Stock',
    traded_at    as 'Date/Time of Trade',
    trade_price  as 'Price/Share',
    cast ( ( ( (trade_price
        - (lag(trade_price, 1)
        over (partition by stock_symbol
            order by traded_at)))
        / trade_price)
    * 100.0) as numeric(5, 2) )
        as '% Price Change vs Previous Price'
from stock_trades
order by 1, 2
```

The query returns these results:

```
Stock   Date/Time of Trade   Price/  % Price Change_vs
symbol                       Share   Previous Price
------  -------------------  -----   -----------------
SQL     2009-07-13 06:07:12  15.84   NULL
SQL     2009-07-13 06:07:15  15.86   0.13
SQL     2009-07-13 06:07:17  15.86   0.00
TST     2009-07-13 06:07:13   5.75   NULL
TST     2009-07-13 06:07:14   5.80   0.87
TST     2009-07-13 06:07:16   5.90   1.72
```

The NULL result in the first and fourth output rows indicates that the LAG function is out of scope for the first row in each of the two partitions. Since there is no previous row to compare to, SAP IQ returns NULL as specified by the <default> variable.

15.3.2.12 Distribution Functions

The inverse distribution functions PERCENTILE_CONT and PERCENTILE_DISC take a percentile value as the function argument and operate on a group of data specified in the WITHIN GROUP clause or operate on the entire data set.

These functions return one value per group. For PERCENTILE_DISC (discrete), the data type of the results is the same as the data type of its ORDER BY item specified in the WITHIN GROUP clause. For PERCENTILE_CONT (continuous), the data type of the results is either numeric, if the ORDER BY item in the WITHIN GROUP clause is a numeric, or double, if the ORDER BY item is an integer or floating point.

The inverse distribution analytical functions require a WITHIN GROUP (ORDER BY) clause. For example:

```
PERCENTILE_CONT ( <expression1> )
WITHIN GROUP ( ORDER BY <expression2> [ ASC | DESC ] )
```

The value of <expression1> must be a constant of numeric data type and range from 0 to 1 (inclusive). If the argument is NULL, then a "wrong argument for percentile" error is returned. If the argument value is less than 0, or greater than 1, then a "data value out of range" error is returned.

The ORDER BY clause, which must be present, specifies the expression on which the percentile function is performed and the order in which the rows are sorted in each group. This ORDER BY clause is used only within the WITHIN GROUP clause and is not an ORDER BY for the SELECT statement.

The WITHIN GROUP clause distributes the query result into an ordered data set from which the function calculates a result.

The value <expression2> is a sort specification that must be a single expression involving a column reference. Multiple expressions are not allowed and no rank analytical functions, set functions, or subqueries are allowed in this sort expression.

The ASC or DESC parameter specifies the ordering sequence as ascending or descending. Ascending order is the default.

Inverse distribution analytical functions are allowed in a subquery, a HAVING clause, a view, or a union. The inverse distribution functions can be used anywhere the simple nonanalytical aggregate functions are used. The inverse distribution functions ignore the NULL value in the data set.

Examples

PERCENTILE_CONT

This example uses the PERCENTILE_CONT function to determine the 10th percentile value for car sales in a region using the following data set:

```
sales       region        dealer_name
```

```
-----       ---------     -----------
900         Northeast     Boston
800         Northeast     Worcester
800         Northeast     Providence
700         Northeast     Lowell
540         Northeast     Natick
500         Northeast     New Haven
450         Northeast     Hartford
800         Northwest     SF
600         Northwest     Seattle
500         Northwest     Portland
400         Northwest     Dublin
500         South         Houston
400         South         Austin
300         South         Dallas
200         South         Dover
```

In the following example query, the `SELECT` statement contains the `PERCENTILE_CONT` function:

```
SELECT region, PERCENTILE_CONT(0.1) WITHIN GROUP
   ( ORDER BY ProductID DESC )
FROM ViewSalesOrdersSales GROUP BY region;
```

The result of the `SELECT` statement lists the 10th percentile value for car sales in a region:

```
region           percentile_cont
---------        ---------------
Canada           601.0
Central          700.0
Eastern          700.0
South            700.0
Western         700.0
```

PERCENTILE_DISC

This example uses the `PERCENTILE_DISC` function to determine the 10th percentile value for car sales in a region, using the following data set:

```
sales       region        dealer_name
-----       ---------     -----------
900         Northeast     Boston
800         Northeast     Worcester
800         Northeast     Providence
700         Northeast     Lowell
540         Northeast     Natick
500         Northeast     New Haven
450         Northeast     Hartford
800         Northwest     SF
600         Northwest     Seattle
500         Northwest     Portland
400         Northwest     Dublin
500         South         Houston
400         South         Austin
300         South         Dallas
200         South         Dover
```

In the following query, the `SELECT` statement contains the `PERCENTILE_DISC` function:

```
SELECT region, PERCENTILE_DISC(0.1) WITHIN GROUP
  (ORDER BY sales DESC )
FROM carSales GROUP BY region;
```

The result of the SELECT statement lists the 10th percentile value for car sales in each region:

```
region          percentile_cont
---------       ---------------
Northeast       900
Northwest       800
South           500
```

See *PERCENTILE_CONT Function [Analytical]* and *PERCENTILE_DISC Function [Analytical]* in *SAP IQ SQL Reference* for complete syntax information.

15.3.3 Numeric Functions

OLAP numeric functions supported by SAP IQ include CEILING (CEIL is an alias), EXP (EXPONENTIAL is an alias), FLOOR, LN (LOG is an alias), SQRT, and WIDTH_BUCKET.

```
numeric value function :: =
 <natural logarithm>
| <exponential function>
| <power function>
| <square root>
| <floor function>
| <ceiling function>
| <width bucket function>
```

Numeric Value Function	Syntax
Natural logarithm	LN (<numeric-expression>)
Exponential function	EXP (<numeric-expression>)
Power function	POWER (<numeric-expression1>, <numeric-expression2>)
Square root	SQRT (<numeric-expression>)
Floor function	FLOOR (<numeric-expression>)
Ceiling function	CEILING (<numeric-expression>)
Width bucket function	WIDTH_BUCKET (<expression>, <min_value>, <max_value>, <num_buckets>)

The semantics of the numeric value functions are:

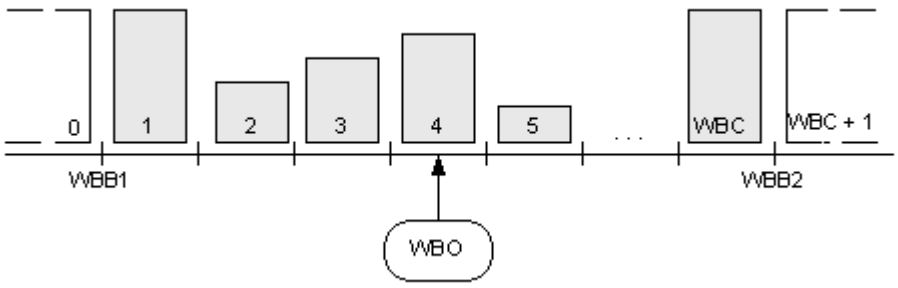
- LN – returns the natural logarithm of the argument value. Raises an error condition if the argument value is 0 or negative. LN is a synonym for LOG.
- EXP – returns the value computed by raising the value of <e> (the base of natural logarithms) to the power specified by the value of the argument.
- POWER – returns the value computed by raising the value of the first argument to the power specified by the value of the second argument. If the first argument is 0 and the second is 0, returns one. If the first argument is 0 and the second is positive, returns 0. If the first argument is 0 and the second argument is negative, raises an exception. If the first argument is negative and the second is not an integer, raises an exception.
- SQRT – returns the square root of the argument value, defined by syntax transformation to "POWER (<expression>, <0.5>)."

- `FLOOR` – returns the integer value nearest to positive infinity that is not greater than the value of the argument.
- `CEILING` – returns the integer value nearest to negative infinity that is not less than the value of the argument. CEIL is a synonym for CEILING.

WIDTH_BUCKET Function

The `WIDTH_BUCKET` function is somewhat more complicated than the other numeric value functions. It accepts four arguments: "live value," two range boundaries, and the number of equal-sized (or as nearly so as possible) partitions into which the range indicated by the boundaries is to be divided. WIDTH_BUCKET returns a number indicating the partition into which the live value should be placed, based on its value as a percentage of the difference between the higher range boundary and the lower boundary. The first partition is partition number one.

To avoid errors when the live value is outside the range of boundaries, live values that are less than the smaller range boundary are placed into an additional first bucket, bucket zero, and live values that are greater than the larger range boundary are placed into an additional last bucket, bucket N+1.



For example, `WIDTH_BUCKET (<14, 5, 30, 5>)` returns 2 because:

- (30-5)/5 is 5, so the range is divided into 5 partitions, each 5 units wide.
- The first bucket represents values from 0.00% to 19.999 ...%; the second represents values from 20.00% to 39.999 ...%; and the fifth bucket represents values from 80.00% to 100.00%.
- The bucket chosen is determined by computing (5*(14-5)/(30-5)) + 1 — one more than the number of buckets times the ratio of the offset of the specified value from the lower value to the range of possible values, which is (5*0/25) + 1, which is 2.8. This value is the range of values for bucket number 2 (2.0 through 2.999 ...), so bucket number 2 is chosen.

WIDTH_BUCKET Example

The following example creates a 10-bucket histogram on the `credit_limit` column for customers in Massachusetts in the sample table and returns the bucket number ("Credit Group") for each customer. Customers with credit limits greater than the maximum value are assigned to the overflow bucket, 11:

> **i** Note
>
> This example is for illustration purposes only and was not generated using the iqdemo database.

```
SELECT customer_id, cust_last_name, credit_limit,
  WIDTH_BUCKET(credit_limit, 100, 5000, 10) "Credit
  Group"
  FROM customers WHERE territory = 'MA'
  ORDER BY "Credit Group";
```

```
CUSTOMER_ID  CUST_LAST_NAME  CREDIT_LIMIT  Credit Group
-----------  --------------  ------------  ------------
825          Dreyfuss                 500             1
826          Barkin                   500             1
853          Palin                    400             1
827          Siegel                   500             1
843          Oates                    700             2
844          Julius                   700             2
835          Eastwood                1200             3
840          Elliott                 1400             3
842          Stern                   1400             3
841          Boyer                   1400             3
837          Stanton                 1200             3
836          Berenger                1200             3
848          Olmos                   1800             4
847           Streep                     5000                 11
```

When the bounds are reversed, the buckets are open-closed intervals. For example: `WIDTH_BUCKET (<credit_limit>, <5000>, <0>, <5>)`. In this example, bucket number 1 is (4000, 5000), bucket number 2 is (3000, 4000), and bucket number 5 is (0, 1000). The overflow bucket is numbered 0 (5000, +infinity), and the underflow bucket is numbered 6 (-infinity, 0).

See *SAP IQ SQL Reference*for complete syntax information for `BIT_LENGTH`, `CEIL`, `CEILING`, `EXP`, `FLOOR`, `LN`, `POWER`, `SQRT`, and `WIDTH_BUCKET`.

15.4 OLAP Rules and Restrictions

SAP IQ provides SQL OLAP functions with rules, restrictions and limitations.

The following provides an overview for the rules and restrictions that govern OLAP functionality.

You can use OLAP functions under these conditions:

- In the `SELECT` list
- In expressions
- As arguments of scalar functions

- In the final `ORDER BY` clause (by using aliases or positional references to OLAP functions elsewhere in the query)

You cannot use OLAP function sunder these conditions:

- In subqueries.
- In the search condition of a `WHERE` clause.
- As arguments for `SET` (aggregate) functions. For example, the following expression is invalid:

```
SUM(RANK() OVER(ORDER BY dollars))
```

- A windowed aggregate cannot be an argument to another unless the inner one was generated within a view or derived table. The same applies to ranking functions.
- Window aggregate and `RANK` functions are not allowed in a `HAVING` clause.
- Window aggregate functions should not specify `DISTINCT`.
- Window function cannot be nested inside of other window functions.
- Inverse distribution functions are not supported with the `OVER` clause.
- Outer references are not allowed in a window definition clause.
- Correlation references are allowed within OLAP functions, but correlated column aliases are not allowed.

Columns referenced by an OLAP function must be grouping columns or aggregate functions from the same query block in which the OLAP function and the `GROUP BY` clause appear. OLAP processing occurs after the grouping and aggregation operations and before the final `ORDER BY` clause is applied; therefore, it must be possible to derive the OLAP expressions from those intermediate results. If there is no `GROUP BY` clause in a query block, OLAP functions can reference other columns in the select list.

SAP IQ Functional Restrictions

The SAP IQ limitations with SQL OLAP functions are:

- User-defined functions in a window frame definition are not supported.
- The constants used in a window frame definition must be unsigned numeric value and should not exceed the value of maximum `BIG INT` 2^{63-1}.
- Window aggregate functions and `RANK` functions cannot be used in `DELETE` and `UPDATE` statements.
- Window aggregate and `RANK` functions are not allowed in subqueries.
- CUME_DIST is currently not supported.
- Grouping sets are currently not supported.
- Correlation and linear regression functions are currently not supported.

15.5 Additional OLAP Examples

This section provides additional examples using the OLAP functions.

Both start and end points of a window may vary as intermediate result rows are processed. For example, computing a cumulative sum involves a window with the start point fixed at the first row of each partition and an end point that slides along the rows of the partition to include the current row.

As another example, both the start and end points of the window can be variable yet define a constant number of rows for the entire partition. Such a construction lets users compose queries that compute moving averages; for example, a SQL query that returns a moving three-day average stock price.

In this section:

Example: Window Functions in Queries [page 397]
This query lists all products shipped in July and August 2005 and the cumulative shipped quantity by shipping date:

Example: Window with Multiple Functions [page 399]
This query defines a single (named) window and computes multiple function results over it:

Example: Calculate Cumulative Sum [page 399]
This query calculates a cumulative sum of salary per department and `ORDER BY start_date`.

Example: Calculate Moving Average [page 400]
This query generates the moving average of sales in three consecutive months. The size of the window frame is three rows: two preceding rows plus the current row. The window slides from the beginning to the end of the partition.

Example: ORDER BY Results [page 400]
In this example, the top `ORDER BY` clause of a query is applied to the final results of a window function. The `ORDER BY` in a window clause is applied to the input data of a window function.

Example: Multiple Aggregate Functions in a Query [page 401]
This example calculates aggregate values against different windows in a query.

Example: Window Frame Comparing ROWS and RANGE [page 401]
This query compares ROWS and RANGE. The data contains duplicate ROWS per the `ORDER BY` clause.

Example: Window Frame Excludes Current Row [page 402]
In this example, you can define the window frame to exclude the current row. The query calculates the sum over four rows, excluding the current row.

Example: Window Frame for RANGE [page 402]
This query illustrates the RANGE window frame. The number of rows used in the summation is variable.

Example: Unbounded Preceding and Unbounded Following [page 403]
In this example, the window frame can include all rows in the partition. The query calculates max(sales) sale over the entire partition (no duplicate rows in a month).

Example: Default Window Frame for RANGE [page 404]
This query illustrates the default window frame for RANGE:

15.5.1 Example: Window Functions in Queries

This query lists all products shipped in July and August 2005 and the cumulative shipped quantity by shipping date:

```
SELECT p.id, p.description, s.quantity, s.shipdate,
```

```
SUM(s.quantity) OVER (PARTITION BY productid
  ORDER BY s.shipdate rows between unbounded preceding and current row)
```

```
   FROM SalesOrderItems s JOIN Products p on(s.ProductID =p.id)
   WHERE s.ShipDate BETWEEN '2001-05-01' and '2001-08-31'
   AND s.quantity > 40 ORDER BY p.id;
```

```
ID   description   quantity   ship_date   sum quantity
---  -----------   --------   ---------   ------------
302  Crew Neck           60  2001-07-02             60
400  Cotton Cap          60  2001-05-26             60
400  Cotton Cap          48  2001-07-05            108
401  Wool cap            48  2001-06-02             48
401  Wool cap            60  2001-06-30            108
401  Wool cap            48  2001-07-09            156
500  Cloth Visor         48  2001-06-21             48
501  Plastic Visor       60  2001-05-03             60
501  Plastic Visor       48  2001-05-18            108
501  Plastic Visor       48  2001-05-25            156
501  Plastic Visor       60  2001-07-07            216
601  Zipped Sweatshirt   60  2001-07-19             60
700  Cotton Shorts       72  2001-05-18             72
700  Cotton Shorts       48  2001-05-31            120
```

In this example, the computation of the `SUM` window function occurs after the join of the two tables and the application of the query's `WHERE` clause. The query uses an inline window specification that specifies that the input rows from the join is processed as follows:

1. Partition (group) the input rows based on the value of the `prod_id` attribute.
2. Within each partition, sort the rows by the `ship_date` attribute.
3. For each row in the partition, evaluate the `SUM()` function over the quantity attribute, using a sliding window consisting of the first (sorted) row of each partition, up to and including the current row.

An alternative construction for the query is to specify the window separate from the functions that use it. This is useful when more than one window function is specified that are based on the same window. In the case of the query using window functions, a construction that uses the window clause (declaring a window identified by cumulative) is as follows:

```
SELECT p.id, p.description, s.quantity, s.shipdate,
   SUM(s.quantity) OVER(cumulative ROWS BETWEEN UNBOUNDED
   PRECEDING and CURRENT ROW ) cumulative FROM SalesOrderItems s
   JOIN Products p On (s.ProductID =p.id)
   WHERE s.shipdate BETWEEN '2001-07-01' and '2001-08-31'
   Window cumulative as (PARTITION BY s.productid
   ORDER BY s.shipdate)ORDER BY p.id;
```

The window clause appears before the `ORDER BY` clause in the query specification. When using a window clause, the following restrictions apply:

- The inline window specification cannot contain a `PARTITION BY` clause.
- The window specified within the window clause cannot contain a window frame clause:

  ```
  <<WINDOW FRAME CLAUSE> ::=
    <WINDOW FRAME UNIT>
    <WINDOW FRAME EXTENT>
  ```

- Either the inline window specification, or the window specification specified in the window clause, can contain a window order clause, but not both:

  ```
  <WINDOW FRAME CLAUSE> ::= <ORDER SPECIFICATION>
  ```

15.5.2 Example: Window with Multiple Functions

This query defines a single (named) window and computes multiple function results over it:

```
SELECT p.ID, p.Description, s.quantity, s.ShipDate, SUM(s.Quantity)
   OVER ws1, MIN(s.quantity) OVER ws1
FROM SalesOrderItems s
JOIN Products p ON (s.ProductID =p.ID)
  WHERE s.ShipDate BETWEEN '2000-01-09' AND'2000-01-17'
  AND s.Quantity > 40 window ws1
  AS(PARTITION BY productid
  ORDER BY shipdate rows between unbounded preceding and current row)
ORDER BY p.id;
```

```
ID   Description   quantity   shipDate     SUM   MIN
---  -----------   --------   -----------  ---   ---
400  Cotton Cap          48    2000-01-09   48    48
401  Wool cap            48    2000-01-09   48    48
500  Cloth Visor         60    2000-01-14   60    60
500  Cloth Visor         60    2000-01-15  120    60
501  Plastic Visor       60    2000-01-14   60    60
```

15.5.3 Example: Calculate Cumulative Sum

This query calculates a cumulative sum of salary per department and `ORDER BY start_date`.

```
SELECT dept_id, start_date, name, salary,
  SUM(salary) OVER (PARTITION BY dept_id ORDER BY
  start_date ROWS BETWEEN UNBOUNDED PRECEDING AND
  CURRENT ROW)
FROM emp1
ORDER BY dept_id, start_date;
```

```
DepartmentID  start_date   name    salary   sum(salary)
```

```
--------     ----------    ----    ------   -----
100          1996-01-01    Anna       18000       18000
100                       1997-01-01    Mike    28000    46000
100                       1998-01-01    Scott   29000    75000
100                       1998-02-01    Antonia 22000    97000
100                       1998-03-12    Adam    25000   122000
100                       1998-12-01    Amy     18000   140000
200                       1998-01-01    Jeff    18000    18000
200                       1998-01-20    Tim     29000    47000
200                       1998-02-01    Jim     22000    69000
200                       1999-01-10    Tom     28000    97000
300                       1998-03-12    Sandy   55000    55000
300                       1998-12-01    Lisa    38000    93000
300                       1999-01-10    Peter   48000   141000
```

15.5.4 Example: Calculate Moving Average

This query generates the moving average of sales in three consecutive months. The size of the window frame is three rows: two preceding rows plus the current row. The window slides from the beginning to the end of the partition.

```
SELECT prod_id, month_num, sales, AVG(sales) OVER
  (PARTITION BY prod_id ORDER BY month_num ROWS
  BETWEEN 2 PRECEDING AND CURRENT ROW)
FROM sale WHERE rep_id = 1
ORDER BY prod_id, month_num;
```

```
prod_id   month_num      sales       avg(sales)
-------   ---------      ------      ----------
10                1        100          100.00
10                2        120          110.00
10                3        100          106.66
10                4        130          116.66
10                5        120          116.66
10                6        110          120.00
20                1         20           20.00
20                2         30           25.00
20                3         25           25.00
20                4         30           28.33
20                5         31           28.66
20                6         20           27.00
30                1         10           10.00
30                2         11           10.50
30                3         12           11.00
30                4          1            8.00
```

15.5.5 Example: ORDER BY Results

In this example, the top `ORDER BY` clause of a query is applied to the final results of a window function. The `ORDER BY` in a window clause is applied to the input data of a window function.

```
SELECT prod_id, month_num, sales, AVG(sales) OVER
  (PARTITION BY prod_id ORDER BY month_num ROWS
  BETWEEN 2 PRECEDING AND CURRENT ROW)
FROM sale WHERE rep_id = 1
ORDER BY prod_id desc, month_num;
```

```
prod_id  month_num       sales   avg(sales)
-------  ---------       -----   ----------
30               1          10       10.00
30               2          11       10.50
30               3          12       11.00
30               4           1        8.00
20               1          20       20.00
20               2          30       25.00
20               3          25       25.00
20               4          30       28.33
20               5          31       28.66
20               6          20       27.00
10               1         100      100.00
10               2         120      110.00
10               3         100      106.66
```

```
10              4         130      116.66
10              5         120      116.66
10              6         110      120.00
```

15.5.6 Example: Multiple Aggregate Functions in a Query

This example calculates aggregate values against different windows in a query.

```
SELECT prod_id, month_num, sales, AVG(sales) OVER
  (WS1 ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS
  CAvg, SUM(sales) OVER(WS1 ROWS BETWEEN UNBOUNDED
  PRECEDING AND CURRENT ROW) AS CSum
FROM sale WHERE rep_id = 1 WINDOW WS1 AS (PARTITION BY
  prod_id
ORDER BY month_num)
ORDER BY prod_id, month_num;
```

```
prod_id  month_num    sales     CAvg     CSum
-------  ---------    -----     ----     ----
10               1      100   110.00      100
10               2      120   106.66      220
10               3      100   116.66      320
10               4      130   116.66      450
10               5      120   120.00      570
10               6      110   115.00      680
20               1       20    25.00       20
20               2       30    25.00       50
20               3       25    28.33       75
20               4       30    28.66      105
20               5       31    27.00      136
20               6       20    25.50      156
30               1       10    10.50       10
30               2       11    11.00       21
30               3       12     8.00       33
30               4        1     6.50       34
```

15.5.7 Example: Window Frame Comparing ROWS and RANGE

This query compares ROWS and RANGE. The data contains duplicate ROWS per the `ORDER BY` clause.

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
  (ws1 RANGE BETWEEN 2 PRECEDING AND CURRENT ROW) AS
    Range_sum, SUM(sales) OVER
  (ws1 ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS
    Row_sum
FROM sale window ws1 AS (PARTITION BY prod_id ORDER BY
  month_num)
ORDER BY prod_id, month_num;
```

```
prod_id  month_num   sales   Range_sum   Row_sum
-------  ---------   -----   ----------  -------
10               1     100         250      100
10               1     150         250      250
10               2     120         370      370
10               3     100         470      370
```

```
10                4     130          350        350
10                5     120          381        350
10                5      31          381        281
10                6     110          391        261
20                1      20           20         20
20                2      30           50         50
20                3      25           75         75
20                4      30           85         85
20                5      31           86         86
20                6      20           81         81
30                1      10           10         10
30                2      11           21         21
30                3      12           33         33
30                4       1           25         24
30                4       1           25         14
```

15.5.8 Example: Window Frame Excludes Current Row

In this example, you can define the window frame to exclude the current row. The query calculates the sum over four rows, excluding the current row.

```
SELECT prod_id, month_num, sales, sum(sales) OVER
  (PARTITION BY prod_id ORDER BY month_num RANGE
  BETWEEN 6 PRECEDING AND 2 PRECEDING)
FROM sale
ORDER BY prod_id, month_num;
```

```
prod_id   month_num       sales    sum(sales)
-------   ---------       -----    ----------
10                1         100        (NULL)
10                1         150        (NULL)
10                2         120        (NULL)
10                3         100          250
10                4         130          370
10                5         120          470
10                5          31          470
10                6         110          600
20                1          20        (NULL)
20                2          30        (NULL)
20                3          25           20
20                4          30           50
20                5          31           75
20                6          20          105
30                1          10        (NULL)
30                2          11        (NULL)
30                3          12           10
30                4           1           21
30                4           1           21
```

15.5.9 Example: Window Frame for RANGE

This query illustrates the RANGE window frame. The number of rows used in the summation is variable.

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
  (PARTITION BY prod_id ORDER BY month_num RANGE
  BETWEEN 1 FOLLOWING AND 3 FOLLOWING)
```

```
FROM sale
ORDER BY prod_id, month_num;
```

```
prod_id   month_num       sales    sum(sales)
-------   ---------       -----    ----------
10                1         100           350
10                1         150           350
10                2         120           381
10                3         100           391
10                4         130           261
10                5         120           110
10                5          31           110
10                6         110        (NULL)
20                1          20            85
20                2          30            86
20                3          25            81
20                4          30            51
20                5          31            20
20                6          20        (NULL)
30                1          10            25
30                2          11            14
30                3          12             2
30                4           1        (NULL)
30                4           1        (NULL)
```

15.5.10 Example: Unbounded Preceding and Unbounded Following

In this example, the window frame can include all rows in the partition. The query calculates max(sales) sale over the entire partition (no duplicate rows in a month).

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
  (PARTITION BY prod_id ORDER BY month_num ROWS
  BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
FROM sale WHERE rep_id = 1
ORDER BY prod_id, month_num;
```

```
prod_id   month_num       sales    SUM(sales)
-------   ---------       -----    ----------
10                1         100           680
10                2         120           680
10                3         100           680
10                4         130           680
10                5         120           680
10                6         110           680
20                1          20           156
20                2          30           156
20                3          25           156
20                4          30           156
20                5          31           156
20                6          20           156
30                1          10            34
30                2          11            34
30                3          12            34
30                4           1            34
```

The query in this example is equivalent to:

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
```

```
  (PARTITION BY prod_id )
FROM sale WHERE rep_id = 1
ORDER BY prod_id, month_num;
```

15.5.11 Example: Default Window Frame for RANGE

This query illustrates the default window frame for RANGE:

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
  (PARTITION BY prod_id ORDER BY month_num)
FROM sale
ORDER BY prod_id, month_num;
```

```
prod_id   month_num       sales    SUM(sales)
-------   ---------       -----    ----------
10                1         100           250
10                1         150           250
10                2         120           370
10                3         100           470
10                4         130           600
10                5         120           751
10                5          31           751
10                6         110           861
20                1          20            20
20                2          30            50
20                3          25            75
20                4          30           105
20                5          31           136
20                6          20           156
30                1          10            10
30                2          11            21
30                3          12            33
30                4           1            35
30                4           1            35
```

The query in this example is equivalent to:

```
SELECT prod_id, month_num, sales, SUM(sales) OVER
  (PARTITION BY prod_id ORDER BY month_num RANGE
  BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
FROM sale
ORDER BY prod_id, month_num;
```

15.6 BNF Grammar for OLAP Functions

The Backus-Naur Form grammar outlines the specific syntactic support for the various ANSI SQL analytic functions, many of which are implemented in SAP IQ.

Grammar Rule	Syntax

1	`<SELECT LIST EXPRESSION> ::=`
	`  <EXPRESSION>`
	`  | <GROUP BY EXPRESSION>`
	`  | <AGGREGATE FUNCTION>`
	`  | <GROUPING FUNCTION>`
	`  | <TABLE COLUMN>`
	`  | <WINDOWED TABLE FUNCTION>`

2	`<QUERY SPECIFICATION> ::=`
	`  <FROM CLAUSE>`
	`  [ <WHERE CLAUSE> ]`
	`  [ <GROUP BY CLAUSE> ]`
	`  [ <HAVING CLAUSE> ]`
	`  [ <WINDOW CLAUSE> ]`
	`[ <ORDER BY CLAUSE> ]`

3	`<ORDER BY CLAUSE> ::=` ` <ORDER SPECIFICATION>`

4	`<GROUPING FUNCTION> ::=`
	`  GROUPING <LEFT PAREN> <GROUP BY EXPRESSION>`
	`  <RIGHT PAREN>`

5	`<WINDOWED TABLE FUNCTION> ::=`
	`  <WINDOWED TABLE FUNCTION TYPE> OVER <WINDOW NAME OR`
	`  SPECIFICATION>`

6	`<WINDOWED TABLE FUNCTION TYPE> ::=`
	`  <RANK FUNCTION TYPE> <LEFT PAREN> <RIGHT PAREN>`
	`  | ROW_NUMBER <LEFT PAREN> <RIGHT PAREN>`
	`  | <WINDOW AGGREGATE FUNCTION>`

7	`<RANK FUNCTION TYPE> ::=` `  RANK | DENSE RANK | PERCENT RANK | CUME_DIST`

8	`<WINDOW AGGREGATE FUNCTION> ::=`
	`  <SIMPLE WINDOW AGGREGATE FUNCTION>`
	`  | <STATISTICAL AGGREGATE FUNCTION>`

Grammar Rule	Syntax

9	`<AGGREGATE FUNCTION> ::=`
	`  <DISTINCT AGGREGATE FUNCTION>`
	`  | <SIMPLE AGGREGATE FUNCTION>`
	`  | <STATISTICAL AGGREGATE FUNCTION>`

10	`<DISTINCT AGGREGATE FUNCTION> ::=`
	`  <BASIC AGGREGATE FUNCTION TYPE> <LEFT PAREN>`
	`  <DISTINCT> <EXPRESSION> <RIGHT PAREN>`
	`  | LIST <LEFT PAREN> DISTINCT <EXPRESSION>`
	`  [ <COMMA> <DELIMITER> ]`
	`  [ <ORDER SPECIFICATION> ] <RIGHT PAREN>`

11	`<BASIC AGGREGATE FUNCTION TYPE> ::=` `  SUM | MAX | MIN | AVG | COUNT`

12	`<SIMPLE AGGREGATE FUNCTION> ::=`
	`  <SIMPLE AGGREGATE FUNCTION TYPE> <LEFT PAREN>`
	`  <EXPRESSION> <RIGHT PAREN>`
	`  | LIST <LEFT PAREN> <EXPRESSION> [ <COMMA>`
	`  <DELIMITER> ]`
	`  [ <ORDER SPECIFICATION> ] <RIGHT PAREN>`

13	`<SIMPLE AGGREGATE FUNCTION TYPE> ::=` ` <SIMPLE WINDOW AGGREGATE FUNCTION TYPE>`

14	`<SIMPLE WINDOW AGGREGATE FUNCTION> ::=`
	`  <SIMPLE WINDOW AGGREGATE FUNCTION TYPE> <LEFT PAREN>`
	`  <EXPRESSION> <RIGHT PAREN>`
	`| GROUPING FUNCTION`

15	`<SIMPLE WINDOW AGGREGATE FUNCTION TYPE> ::=`
	`  <BASIC AGGREGATE FUNCTION TYPE>`
	`  | STDDEV | STDDEV_POP | STDDEV_SAMP`
	`  | VARIANCE | VARIANCE_POP | VARIANCE_SAMP`

16	`<STATISTICAL AGGREGATE FUNCTION> ::=`
	`  <STATISTICAL AGGREGATE FUNCTION TYPE> <LEFT PAREN>`
	`  <DEPENDENT EXPRESSION> <COMMA> <INDEPENDENT`
	`  EXPRESSION> <RIGHT PAREN>`

17	`<STATISTICAL AGGREGATE FUNCTION TYPE> ::=`
	`  CORR | COVAR_POP | COVAR_SAMP | REGR_R2 |`
	`  REGR_INTERCEPT | REGR_COUNT | REGR_SLOPE |`
	`  REGR_SXX | REGR_SXY | REGR_SYY | REGR_AVGY |`
	`  REGR_AVGX`

18	`<WINDOW NAME OR SPECIFICATION> ::=` `  <WINDOW NAME> | <IN-LINE WINDOW SPECIFICATION>`

Grammar Rule	Syntax
19	`<WINDOW NAME> ::= <IDENTIFIER>`
20	`<IN-LINE WINDOW SPECIFICATION> ::=` ` <WINDOW SPECIFICATION>`
21	`<WINDOW CLAUSE> ::=` ` <WINDOW WINDOW DEFINITION LIST>`

22	`<WINDOW DEFINITION LIST> ::=`
	`  <WINDOW DEFINITION> [ { <COMMA> <WINDOW DEFINITION>`
	`  } . . . ]`

23	`<WINDOW DEFINITION> ::=` `  <NEW WINDOW NAME> AS <WINDOW SPECIFICATION>`
24	`<NEW WINDOW NAME> ::= <WINDOW NAME>`

25	`<WINDOW SPECIFICATION> ::=`
	`  <LEFT PAREN> <WINDOW SPECIFICATION> <DETAILS> <RIGHT`
	`  PAREN>`

26	`<WINDOW SPECIFICATION DETAILS> ::=`
	`  [ <EXISTING WINDOW NAME> ]`
	`  [ <WINDOW PARTITION CLAUSE> ]`
	`  [ <WINDOW ORDER CLAUSE> ]`
	`  [ <WINDOW FRAME CLAUSE> ]`

27	`<EXISTING WINDOW NAME> ::=` ` <WINDOW NAME>`
28	`<WINDOW PARTITION CLAUSE> ::=` `  PARTITION BY <WINDOW PARTITION EXPRESSION LIST>`

29	`<WINDOW PARTITION EXPRESSION LIST> ::=`
	`  <WINDOW PARTITION EXPRESSION>`
	`  [ { <COMMA> <WINDOW PARTITION EXPRESSION> } . . . ]`

30	`<WINDOW PARTITION EXPRESSION> ::=` ` <EXPRESSION>`
31	`<WINDOW ORDER CLAUSE> ::=` ` <ORDER SPECIFICATION>`

Grammar Rule	Syntax

32	`<WINDOW FRAME CLAUSE> ::=`
	`  <WINDOW FRAME UNIT>`
	`  <WINDOW FRAME EXTENT>`

33	`<WINDOW FRAME UNIT> ::=` ` ROWS | RANGE`
34	`<WINDOW FRAME EXTENT> ::=` ` <WINDOW FRAME START> | <WINDOW FRAME BETWEEN>`

35	`<WINDOW FRAME START> ::=`
	`  UNBOUNDED PRECEDING`
	`  | <WINDOW FRAME PRECEDING>`
	`  | CURRENT ROW`

36	`<WINDOW FRAME PRECEDING> ::=` ` <UNSIGNED VALUE SPECIFICATION> PRECEDING`

37	`<WINDOW FRAME BETWEEN> ::=`
	`  BETWEEN <WINDOW FRAME BOUND 1> AND <WINDOW FRAME`
	`  BOUND 2>`

38	`<WINDOW FRAME BOUND 1> ::=` ` <WINDOW FRAME BOUND>`
39	`<WINDOW FRAME BOUND 2> ::=` ` <WINDOW FRAME BOUND>`

40	`<WINDOW FRAME BOUND> ::=`
	`  <WINDOW FRAME START>`
	`  | UNBOUNDED FOLLOWING`
	`  | <WINDOW FRAME FOLLOWING>`

41	`<WINDOW FRAME FOLLOWING> ::=` ` <UNSIGNED VALUE SPECIFICATION> FOLLOWING`
42	`<GROUP BY EXPRESSION> ::=` ` <EXPRESSION>`

43	`<SIMPLE GROUP BY TERM> ::=`
	`  <GROUP BY EXPRESSION>`
	`  | <LEFT PAREN> <GROUP BY EXPRESSION> <RIGHT PAREN>`
	`  | <LEFT PAREN> <RIGHT PAREN>`

Grammar Rule	Syntax

44	`<SIMPLE GROUP BY TERM LIST> ::=`
	`  <SIMPLE GROUP BY TERM> [ { <COMMA> <SIMPLE GROUP BY`
	`  TERM> } . . . ]`

45	`<COMPOSITE GROUP BY TERM> ::=`
	`  <LEFT PAREN> <SIMPLE GROUP BY TERM>`
	`  [ { <COMMA> <SIMPLE GROUP BY TERM> } . . . ]`
	`  <RIGHT PAREN>`

46	`<ROLLUP TERM><COMPOSITE GROUP BY TERM>`
47	`<CUBE TERM> ::=` ` CUBE <COMPOSITE GROUP BY TERM>`

48	`<GROUP BY TERM> ::=`
	`  <SIMPLE GROUP BY TERM>`
	`  | <COMPOSITE GROUP BY TERM>`
	`  | <ROLLUP TERM>`
	`  | <CUBE TERM>`

49	`<GROUP BY TERM LIST> ::=` `  <GROUP BY TERM> [ { <COMMA> <GROUP BY TERM> } … ]`
50	`<GROUP BY CLAUSE> ::=` ` GROUP BY <GROUPING SPECIFICATION>`

51	`<GROUPING SPECIFICATION> ::=`
	`  <GROUP BY TERM LIST>`
	`  | <SIMPLE GROUP BY TERM LIST> WITH ROLLUP`
	`  | <SIMPLE GROUP BY TERM LIST> WITH CUBE`

52	Not supported.

53	`<ORDER SPECIFICATION> ::=`
	` ORDER BY <SORT SPECIFICATION LIST>`
	`  <SORT SPECIFICATION LIST> ::= <SORT SPECIFICATION>`
	`  [ { <COMMA> <SORT SPECIFICATION> } . . . ]`
	`  <SORT SPECIFICATION> ::= <SORT KEY>`
	`  [ <ORDERING SPECIFICATION> ] [ <NULL ORDERING> ]`
	`  <SORT KEY> ::= <VALUE EXPRESSION>`
	`  <ORDERING SPECIFICATION> ::= ASC | DESC`
	`  <NULL ORDERING> := NULLS FIRST | NULLS LAST`

16 Troubleshooting Hints

SAP IQ provides many resources for addressing problems.

In this section:

Online Resources [page 410]
SAP IQ online resources are just a click away. Learn about the product, connect to the community, or scan the video catalog. Find what you want here.

Solutions for Specific Conditions [page 412]
More information may be needed to diagnose and resolve certain issues. You can use diagnostic tools to diagnose various conditions

Troubleshooting Network Communications [page 440]
Network software involves several different components, increasing the likelihood of issues requiring troubleshooting.

Diagnostic Tools [page 444]
Several tools help you diagnose various conditions.

Reporting Problems to Technical Support [page 459]
If you cannot resolve a problem using the manuals or online help, the designated person should contact SAP Technical Support or the SAP subsidiary in your area.

Checklist: Information for Technical Support [page 463]
You can run the `getiqinfo` script to collect information.

16.1 Online Resources

SAP IQ online resources are just a click away. Learn about the product, connect to the community, or scan the video catalog. Find what you want here.

16.1.1 Community

- SAP IQ Developer Center
- SAP IQ Community Network (SCN)
- SAP Business Warehouse Community Network (SCN)

16.1.2 Documentation

- SAP IQ Help Portal
  Links to SAP IQ documentation in HTML and PDF formats.
- Hardware Sizing Guide
  Highlights the main areas of concern for sizing CPU, memory, and storage forSAP IQ.
- Best Practices Guide
  Get the best performance from your software, identify potential issues, troubleshoot problems.

16.1.3 Video Catalog

SAP IQ includes an extensive video library.

See *SAP IQ "How-To" Video Tutorials* to view step-by-step tutorial videos covering the setup, configuration, and administration of your SAP IQ server, designed for you by our SAP IQ experts.

16.1.4 Demos

- Utilize Big Data for interesting insights into your business
  Demonstrates data and query federation through invocation of in-database Java functions from SQL that extract data from the Hadoop file system (data federation) or invokes Hadoop MapReduce jobs (query federation) and combines the results with data in the SAP IQ database.
- Want to analyze your Big Data without a big hassle?
  Demonstrates how SAP IQ can execute massively parallel MapReduce - like processing completely within the database without a separate Big Data processing environment like Hadoop.
- Want to know what your customers are thinking?
  Demonstrates how SAP BusinessObjects EIM and BI tools in combination with ISYS Document Pre-filters and SAP IQ full text search can perform sentiment analysis on customer reviews about automobiles.
- Prevent Customer Churn By Hearing What Your Customers Are Saying
  Demonstrates how SAP IQ BusinessObjects EIM and BI tools in combination with ISYS Document Pre-filters and SAP IQ full text search can perform sentiment analysis of users commenting about their respective telephone network service on Facebook and Twitter.
- The Power of Predictive Analytics
  Demonstrates social network analysis using KXEN InfiniteInsight Social running on top of SAP IQ that can churn through large amounts of data at high speeds.

16.1.5 Near-Line Storage Solution

- SAP BW Near-Line Storage Solution (NLS) Based on SAP IQ

Describes the adapter for SAP IQ as a near-line solution delivered with Business Warehouse (BW) on SAP HANA.

- Installing and Configuring SAP IQ as Extended Storage for SAP HANA
  Tells you how to install and configure SAP IQ as the extended store for SAP Business Warehouse on HANA.
- SAP First Guidance: Near-Line Storage (NLS) Solution with SAP IQ
  Includes prerequisites, installation, and maintenance tasks.

16.1.6 SAP IQ and Hadoop

- Set Up Hortonworks Hadoop with SAP IQ
  Setup required to use SAP IQ with the Hortonworks distribution of the Hive server and HiveQL.
- Set Up Intel Hadoop with SAP IQ
  Setup required to use IQ with the Intel distribution of the Hive server and HiveQL.
- Hadoop to SAP IQ through FIFO Pipe:
  Describes a method to push data from the Hadoop file system (HDFS) to a FIFO pipe, then load the data into a global temporary table with the LOAD TABLE statement.

16.1.7 Whitepapers

- Distributed Query Processing in SAP IQ on VMware VSphere and Virtual SAN
- Building a 12 Petabyte Data Warehouse
- Evaluation of SAP IQ 16 Multiplex Performance with Various Storage and Network Technologies
- Bay Storage Technology brings I/O performance improvements to SAP IQ at low cost
- Boost the Value of Big Data with SAP BI and SAP IQ

16.2 Solutions for Specific Conditions

More information may be needed to diagnose and resolve certain issues. You can use diagnostic tools to diagnose various conditions

In this section:

Decision Flow for Server Recovery and Database Repair [page 413]
You may experience trouble starting a server or database, or connecting to or verifying a database.

Server Operational Issues [page 414]
Issues that may affect server operation include startup, shutdown, unresponsiveness, and abnormal termination.

Database Connection Issues [page 423]
You may encounter issues while attempting to connect to a database.

Interactive SQL (dbisql) Issues [page 425]
You may encounter issues while using `dbisql`.

Resource Issues [page 425]
Resource issues may include insufficient disk space, insufficient number of threads, thread stack overflow, and unused system resources.

Processing Issues [page 434]
Processing issues may be related to loads, queries, indexes, and table access.

Performance Issues [page 439]
Certain settings may impact performance.

16.2.1 Decision Flow for Server Recovery and Database Repair

You may experience trouble starting a server or database, or connecting to or verifying a database.

Procedure

1. Does the server start?

Option	Description
Yes	Go to step 2.
No	See Server Operational Issues [page 414]. If you cannot start the server after following these suggestions in this section, see *Starting a Server in Forced Recovery Mode* in *SAP IQ Administration: Backup, Restore, and Data Recovery* and start the server in forced recovery mode. If the server does not start in forced recovery mode, call Technical Support. You may need to restore the database from backup.

2. Can you connect to the database?

Option	Description
Yes	If you can connect to the database and you previously started the server in forced recovery, see *Analysis of Allocation Problems* in *SAP IQ Administration: Backup, Restore, and Data Recovery* for information on verifying database allocation and recovering leaked blocks. If you can connect to the database, but suspect the database may be inconsistent, see *Database Verification* in *SAP IQ Administration: Backup, Restore, and Data Recovery* for information on checking the consistency of your database.
No	If you cannot connect to the database, see Database Connection Issues [page 423] for troubleshooting suggestions.

3. The server is running and you can connect, but you want to verify the consistency of your database.

If you previously started the server with forced recovery or you suspect database inconsistency, run DBCC checks to validate the database. See *Database Verification* for information on checking both index consistency and database allocation.

4. The server is running, you can connect, you have run DBCC checks, and you need to repair the index inconsistencies or allocation problems detected by DBCC.

   If `sp_iqcheckdb` reports errors in the Index Summary and Index Statistics sections of the results, see *Index Error Repair* in *SAP IQ Administration: Backup, Restore, and Data Recovery* for the procedure to repair index problems using DBCC.

   If `sp_iqcheckdb` reports errors in the Allocation Summary and Allocation Statistics sections of the results, see *Repairing Allocation Problems using DBCC* in *SAP IQ Administration: Backup, Restore, and Data Recovery* for the procedure to repair allocation problems using DBCC.

16.2.2 Server Operational Issues

Issues that may affect server operation include startup, shutdown, unresponsiveness, and abnormal termination.

In this section:

SAP IQ Will Not Start [page 414]
If there is a problem starting the server, `start_iq` returns a non zero value.

SAP IQ Stops Processing or Stops Responding [page 419]
You can detect the cause of server unresponsiveness by looking in the SAP IQ message file.

System or SAP IQ Failure [page 421]
You can detect the cause of system or SAP IQ failures by looking in the SAP IQ message file.

Server Fails to Shut Down [page 422]
To shut down the server, run the `dbstop` utility or `stop_iq`, type `q` in the server window on UNIX, or click **Shutdown** on the server window on Windows.

16.2.2.1 SAP IQ Will Not Start

If there is a problem starting the server, `start_iq` returns a non zero value.

If you did not specify a log file after the `-o` switch on startup, SAP IQ writes the error to the first one of the following that is defined:

UNIX	• `$IQDIR16/logfiles/<servername>.nnnn.stderr` • `$IQDIR16/logfiles/<servername>.nnnn.srvlog` • The Systems applications log file
Windows	• `%IQDIR16%\logfiles\<servername>.nnnn.stderr` • `%IQDIR16%\logfiles\<servername>.nnnn.srvlog` • The Systems applications log file

There are several possible causes.

In this section:

Transaction Log File Does Not Match the Database [page 415]
Messages appear in the server log file (`.srvlog`) and in the window where you are starting the server:

Server Cannot Find the Transaction Log [page 416]
If the server fails to start because it cannot find the transaction log, messages appear in the server log file.

Server Name Is Not Unique on Your Network [page 416]
If multiple servers on your system have the same name, messages appear in the server log file (`*.srvlog` or the name specified in the `-o` startup option) when you attempt to start the server using `start_iq`.

Log File Has Illegal Name [page 416]
If you specified a separate request-level logging file, but the file name is an illegal identifier, errors result on server startup.

Server Port Number Is Not Unique on the Machine [page 417]
If an SAP IQ server is running and you attempt to start another SAP IQ server on the same machine using the same port number, messages appear in the server log file (`*.srvlog`).

Server Started with an Incorrect Path [page 417]
When you start a new multiplex server, the database file path must match the database file path specified when creating that server.

Environment Variables Not Set Correctly [page 418]
If your database configuration file parameters differ from those used by `start_iq`, make sure the correct parameters are used to start the server.

You Cannot Run start_iq [page 418]
If you cannot run the `start_iq` command and you normally use a configuration file or other command line switches, try starting the server using only `start_iq` with the server name and database name.

16.2.2.1.1 Transaction Log File Does Not Match the Database

Messages appear in the server log file (`.srvlog`) and in the window where you are starting the server:

```
Starting database "dbname" (/dbdir/dbname.db)
at Fri Apr 27 2009 10:53 Transaction log: dbname.log
Error: Cannot open transaction log file
-- Can't use log file "dbname.log" since the database
file has been used more recently
Cannot open transaction log file
-- Can't use log file "dbname.log" since the database
file has been used more recently
Database server stopped at Fri Apr 27 2009 10:53
```

If these errors are reported when you are starting the server, verify that the server is using the correct transaction log file. If you cannot find the correct transaction log file, the safest way to recover from this situation is to restore from the last valid backup.

If you cannot find the correct transaction log and restoring from backup is not an option, perform an emergency recovery without a transaction log.

16.2.2.1.2 Server Cannot Find the Transaction Log

If the server fails to start because it cannot find the transaction log, messages appear in the server log file.

```
Transaction log: /dbdir/dbname.log...
Error: Cannot open transaction log file
-- No such file or directory
Cannot open transaction log file
-- No such file or directory
```

If this error is reported when you attempt to start the server, find the transaction log file and copy the file to the same directory as the database `.db` file. If you cannot find the correct transaction log file, restore from the last valid backup.

If no other option for starting the server is available, you may be able to start the server using the emergency recovery `-f` option. Contact SAP Technical Support for assistance, if necessary.

> ⚠ Caution
>
> This procedure is highly risky and is not recommended except in extreme cases.

16.2.2.1.3 Server Name Is Not Unique on Your Network

If multiple servers on your system have the same name, messages appear in the server log file (`*.srvlog` or the name specified in the `-o` startup option) when you attempt to start the server using `start_iq`.

```
DBSPAWN ERROR:  -85
Communication error
```

If you see these errors in the server log file and the server does not start, try to start the server using the `iqsrv16` command. The `iqsrv16` command returns a more specific error message:

```
A database server with that name has already started
```

Once you have verified that the problem is a duplicate server name on your network, start the server with a name that is different from the names of servers that are already running.

16.2.2.1.4 Log File Has Illegal Name

If you specified a separate request-level logging file, but the file name is an illegal identifier, errors result on server startup.

```
Naming conflict: "iqdemo" --
```

```
aborting
Database naming conflict --
aborting startup
```

These errors may indicate a space in the file path specified on the `-zo` option.

Specify the `-zo` option again and enclose any file name that contains a space within quotation marks.

16.2.2.1.5 Server Port Number Is Not Unique on the Machine

If an SAP IQ server is running and you attempt to start another SAP IQ server on the same machine using the same port number, messages appear in the server log file (`*.srvlog`).

```
Trying to start TCPIP link ...
TCPIP communication link not started
Unable to initialize requested communication links
...
DBSPAWN ERROR:  -85
Communication error
Server failed to start
```

If you see these messages in the server log file and the server does not start, run the `stop_iq` command (UNIX) to display the names and port numbers of SAP IQ servers already running on the machine. Then try to start your server, specifying either a port number that is not in use or no port number. When you start a server and do not provide a port number (and the default port number is already in use), SAP IQ generates an available port number.

You see these messages in the server log file when you start the server without specifying a port number:

```
Trying to start TCPIP link ...
Unable to start on default port; starting on port
49152 instead
TCPIP link started successfully
Now accepting requests
...
Server started successfully
```

16.2.2.1.6 Server Started with an Incorrect Path

When you start a new multiplex server, the database file path must match the database file path specified when creating that server.

If you use the wrong path, server startup fails, and writes these messages in the server log file (`*.srvlog`):

```
E. 08/18 07:22:19. MPX: server myserver
has been started with an incorrect catalog path
(expected path: /work/IQ-16_1/demo/mympx/iqdemo.db).
-- (st_database.cxx 7883)
I. 08/18 07:22:19. Database server shutdown due
to startup error
DBSPAWN ERROR:  -82
Unable to start specified database: autostarting
database failed
```

If you see these messages, restart the server with the expected path. If you plan to use UNIX soft (symbolic) links for server paths, you must create the soft link before you run `CREATE MULTIPLEX SERVER`.

16.2.2.1.7 Environment Variables Not Set Correctly

If your database configuration file parameters differ from those used by `start_iq`, make sure the correct parameters are used to start the server.

16.2.2.1.8 You Cannot Run start_iq

If you cannot run the `start_iq` command and you normally use a configuration file or other command line switches, try starting the server using only `start_iq` with the server name and database name.

If the server starts with this simple command, then the problem is probably caused by one or more of the switches or parameters entered on the command line or in the configuration file. Try to isolate which parameter or switch is preventing the server from starting.

If the server does not start with the most basic `start_iq` command, try starting the `iqdemo` demo database using your configuration file and command line switches. If the server starts with the `iqdemo` database, there may be a problem with your database.

If you still cannot run the `start_iq` command, use the `iqsrv16` command.

> **i Note**
>
> Use `iqsrv16` only for troubleshooting server start up errors. Always use `start_iq` to start SAP IQ servers.

Before running `iqsrv16`, you must perform the following tasks (which `start_iq` normally does for you):

- Remove all limits, and then set limits on the stack size and descriptors. To do so, go to the C shell and issue these commands:

  ```
  % unlimit
  % limit stacksize 8192
  % limit descriptors 4096
  ```

  > **i Note**
  >
  > `unlimit` affects soft limits only. You must change any hard limits by setting kernel parameters.

- Set all server options appropriately for your platform. See the *SAP IQ Installation and Update Guide*.
- Add the path `$SYBASE/OCS-16_0/lib` (for UNIX) or `%SYBASE%/OCS-16_0\lib` (for Windows) to the environment to load the engine and required libraries before you invoke `iqsrv16`. Put this path in the environment only during testing, as follows:
  On AIX:

  ```
  % setenv LIBPATH "${LIBPATH}:{SYBASE}/OCS-16_0/lib"
  ```

On other UNIX/LINUX platforms:

```
% setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:${SYBASE}/OCS-16_0/lib"
```

For any database created with a relative path name, you must start the database server from the directory where the database is located.

Note what directory you are in when you start the server. The server startup directory determines the location of any new database files you create with relative path names. If you start the server in a different directory, SAP IQ cannot find those database files.

Any server startup scripts should change to a known location before issuing the server startup command.

The syntax for `iqsrv16` is:

```
iqsrv16 -n <server-name> -gm <number>
[ <other-server-switches> ] [ <database-file> [ <database-switches> ] ]
```

> **Note**
>
> On the `iqsrv16` command line, the last option specified takes precedence, so to override your configuration file, list any options you want to change after the configuration file name. For example:
>
> ```
> iqsrv16 @iqdemo.cfg -x 'tcpip{port=1870}' iqdemo
> ```
>
> The `-x` parameter here overrides connection information in the `iqdemo.cfg` file.

If the server fails to start when you run the `iqsrv16` command, then attempt to start again using the `iqsrv16` utility with minimal switches and parameters. For example:

```
iqsrv16 -n <servername> <dbname>.db -c 32m
-gd all -gl all
```

If the server starts with the minimum parameters and switches, then one of the parameters or switches normally used to start the server may be causing a problem. Try to isolate which parameter or switch is preventing the server from starting.

When you start the server with the `iqsrv16` command, it does not run in the background, and messages do not automatically go to the server log. However, if you include the `-o` file name server switch, messages are sent to the named file in addition to the server window.

16.2.2.2 SAP IQ Stops Processing or Stops Responding

You can detect the cause of server unresponsiveness by looking in the SAP IQ message file.

Context

The most common causes of server unresponsiveness include:

- Insufficient disk space
- Insufficient room in main or temp buffer cache

Procedure

- If your server seems to be prone to unresponsiveness, either while processing or during shutdown, use the `start_iq` command line option `-z` and the SAP IQ database option `QUERY_PLAN = 'ON'` to log useful information in the SAP IQ message (`.iqmsg`) and server log (`.srvlog`) files.
- In addition to logging this information, there are other steps you can take to determine the cause of the problem:
  - Check both the SAP IQ message file and the server log file for `You have run out of space...` messages. If you have run out of IQ main store or IQ temporary store, add the appropriate dbspace with the `CREATE DBSPACE` command.
    Setting the database options `MAIN_RESERVED_DBSPACE_MB` and `TEMP_RESERVED_DB_SPACE_MB` to large enough values to handle running out of space during a DDL `COMMIT` or `CHECKPOINT` is also important. A few hundred MB should be enough, but you can set these options higher for a large database.
  - Determine if the SAP IQ server process (`iqsrv16`) is consuming CPU cycles by monitoring the CPU usage for a few minutes at the operating system level. Record this information. If the CPU usage changes, then the SAP IQ server process should be processing normally.
    If the SAP IQ server CPU usage is normal, you can examine what the server is doing, that is, what statement the server is currently executing.
  - If there are no out of space indications, use Interactive SQL on a new or existing connection to gather the following information, in the specified order.

Command	Informational Purpose
SELECT db_name()	Database name
CHECKPOINT	Checkpoint can succeed
sa_conn_properties ># sa_conn_properties.out	Connection information
sa_conn_info ># sa_conn_info.out	Connection information
sa_db_properties ># sa_db_properties.out	Database property information
sa_eng_properties ># sa_eng_properties.out	Server property information
sp_iqstatus ># sp_iqstatus.out	Database status information
sp_iqconnection ># sp_iqconnection.out	Connection information
sp_iqtransaction ># sp_iqtransaction.out	Transaction information

  - If you cannot resolve the issue, contact SAP Technical Support for assistance. They can use the information you have just gathered to help diagnose the problem.
  - When the server is unresponsive, generate a stack trace for each SAP IQ thread by creating a file named `DumpAllThreads` or `dumpallthreads` in the `$IQDIR16/logfiles` directory (the `%ALLUSERSPROFILE\%\SAP\logfiles` folder on Windows 64 platforms, `C:\ProgramData\SAPIQ\logfiles` for Vista 64).

Starting SAP IQ as recommended, using the Program Manager or `start_iq` command, sets the `<IQDIR16>` variable automatically. If the `<IQDIR16>` variable is not set, create the `DumpAllThreads` file in the directory in which `iqsrv16` was started.
The SAP IQ server detects the presence of the `DumpAllThreads` file and writes a stack trace for each IQ thread in the stack trace file `stktrc-YYYYMMDD-HHNNSS_#.iq`. After the stack traces are written to the stack trace file, the `DumpAllThreads` file is deleted.
This stack trace information can be used by SAP Technical Support to help diagnose the problem.

- If you can connect to the database, run the `IQ UTILITIES` buffer cache monitor on the main and temp (private) buffer caches for ten minutes with a ten-second interval:
  1. Connect to the database or use the existing connection.
  2. `CREATE TABLE #dummy_monitor(c1 INT);`
  3. `IQ UTILITIES MAIN INTO #dummy_monitor START MONITOR '-append -debug -interval 10 -file_suffix iqdbgmon';`
  4. `IQ UTILITIES PRIVATE INTO #dummy_monitor START MONITOR '-append -debug -interval 10 -file_suffix iqdbgmon';`
     Let the process run for 10 minutes, then stop the buffer cache monitor:
  5. `IQ UTILITIES MAIN INTO #dummy_monitor STOP MONITOR;`
  6. `IQ UTILITIES PRIVATE INTO #dummy_monitor STOP MONITOR;`
- Check near the end of the SAP IQ message file for the message `Resource count 0`, which may be followed by an `Open Cursor` message. These messages indicate a resource depletion, which can cause a deadlock. The immediate solution is to reduce the number of active connections using Ctrl+C or the `DROP CONNECTION` command.
  The long-term solution for avoiding deadlocks due to resource depletion is one or a combination of:
  - Restricting the number of users on the server by reducing the value of the `-gm` server startup option
  - Adding another secondary server to a multiplex
  - Increasing the processing capacity of the hardware by adding CPUs

16.2.2.3 System or SAP IQ Failure

You can detect the cause of system or SAP IQ failures by looking in the SAP IQ message file.

Context

There may be various possible causes.

Procedure

- Copy or rename the message log file (`dbname.iqmsg`) before trying to restart the database. This ensures that any useful information in the file is not lost.

- On UNIX, send a copy of the stack trace to SAP Technical Support. The stack trace should be in the directory where you started the database server, in a file named `stktrc-YYYYMMDD-HHNNSS_#.iq`. If the database was open when the failure occurred, the stack trace should also be in the SAP IQ message log (default name `dbname.iqmsg`). This information helps SAP Technical Support determine why the failure occurred.
- Restart the server with the `start_iq` command. When the database restarts, recovery occurs automatically.
- Try to start the server without starting a database. If you can start the server but not the database, check that database parameters are specified correctly on the startup line and in the connection profile.
- If you query catalog store tables extensively, restart the server and make sure that the `TEMP_SPACE_LIMIT_CHECK` option is on. With this option setting, if a connection exceeds its quota of catalog store temporary file space, it receives a non fatal error.

16.2.2.4 Server Fails to Shut Down

To shut down the server, run the `dbstop` utility or `stop_iq`, type `q` in the server window on UNIX, or click **Shutdown** on the server window on Windows.

Possible Causes

Various.

Actions

Perform these actions if the server fails to shut down.

On UNIX systems:

1. If your system is a multiplex, log into the coordinator. Otherwise, continue with the next step.
2. Capture `ps` operating system utility output, so you can submit this output to Technical Support. On Sun Solaris, two different `ps` options are available. Use both.

   ```
   ps -aAdeflcj|egrep "PPID|iqsrv16"
   ```

   ```
   /usr/ucb/ps -awwwlx|egrep "PPID|iqsrv16"
   ```
3. Try to kill the process at the operating system level to generate a core dump.

   ```
   kill -1 <pid>
   ```

   A small core file is created in the directory where `start_iq` was run. If you can kill the server process in this way, skip to step 7. Make sure that you allow time for the command to work before you try another kill command.

4. If `kill -1` (SIGHUP) fails, try `kill -6` (SIGABRT).

   ```
   kill -6 <pid>
   ```

   A small core file is created in the directory where `start_iq` was run. If you can kill the server process in this way, skip to step 7.
5. If the server process still does not exit, capture `ps` output as in step 1. Retain the output from both times you run `ps` (before and after trying to kill the process). Then kill the process with a stronger signal. Make sure that you allow time for the command in Step 4 to work before you try another kill command. The `kill-9` command is an operating system request to destroy the process that bypasses all IQ diagnostic and shutdown logic. It should never become the habitual means of shutdown, and should only be used as a last resort:

   ```
   kill -9 <pid>
   ```
6. If this method does not cause the process to exit, capture yet another set of `ps` output, then restart your system.
7. Submit all `ps` output, the core file (if generated in step 2), and the stack trace in `stktrc-YYYYMMDD-HHNNSS_#.iq` to Technical Support.

On Windows systems:

1. If your system is a multiplex, log into the coordinator. Otherwise, continue with the next step.
2. Start the Task Manager by right-clicking the Task Bar and clicking **Task Manager**.
3. In the Processes tab, select **iqsrv16.exe**, then click the **End Process** button to stop the database server.
4. If necessary, restart Windows.

16.2.3 Database Connection Issues

You may encounter issues while attempting to connect to a database.

In this section:

Cannot Connect to a Database [page 423]
You may experience problems connecting to a database.

16.2.3.1 Cannot Connect to a Database

You may experience problems connecting to a database.

Possible Causes

- Data source is not defined, or is defined incorrectly. Try connecting again with the correct user ID and password.

A data source is a set of connection parameters, stored in the registry (on Windows) or in a file (Windows and UNIX).

- An incorrect user name or password is specified.
- User may not have permission to use the database.
- You are connecting over TDS (for example, using jConnect) and the user ID or password is longer than 30 bytes. You see:

```
Invalid user ID or password
CT-LIBRARY error:
ct_connect(): protocol specific layer:
external error: The attempt to connect to the server failed.
```

- You provide an incorrect database file name. Try connecting again with the correct database file name. You must supply the `DBF` parameter and the database file name to connect when you use Interactive SQL and you have restored the database from backup while connected to `utility_db`.
- Database files may be missing. The files `dbname.db`, `dbname.iq`, and `dbname.iqmsg` (where `dbname` is the name of the database) must all exist.
- A limit on the number of connections or other DBA-defined login restrictions may be exceeded.
- You have run out of disk space. Check the SAP IQ message file for messages related to disk space.
- The server name specified is incorrect. Check the name of the server and try connecting again with the correct server name.
- The server machine name or address has changed.
- When connecting from a client for the first time and the server name is not specified, providing the wrong port number can cause a failure to connect to the database. The error messages returned are:

```
Could not connect to the database.
Database server not found.
```

  When connecting from Interactive SQL, ensure that the name in the **Server Name** field is spelled correctly that the network options on the network tab are correct, and that the database server has been started. Either provide the server name when connecting, or use the correct port number. To determine the server name and the number of the port on which the server is listening, run `stop_iq` (UNIX), which displays this information.
- Port number may be out of correct range or in use by another process.
- If you receive either of the following messages when trying to start the client, the client cannot find the database server on the network:
  - `Unable to start — server not found`
  - `Database server not running.`

  The connection string may be incorrect or the server name cache may contain incorrect or old connection information. For example, if the server is started with a different port number, even if the client application specifies the new port number at connect time, the connection information is still taken from the server name cache.
- You specified a character set in the CharSet connection parameter and tried to connect to a server that does not support that character set.
  Try reconnecting without specifying CharSet. If the client's local character set is unsupported by the server, the connection succeeds, but with a warning that the character set is not supported.

Action

If you suspect that you cannot connect because there is a problem with the database, look in the `dbname.iqmsg` file to determine where the problem occurred.

`Open Database Completed` indicates that the database opened without error and the problem is related to the clients connecting. If the message does not appear, then the database may have failed while opening or recovering.

16.2.4 Interactive SQL (dbisql) Issues

You may encounter issues while using `dbisql`.

In this section:

Directories Remain After Exiting dbisql [page 425]
This issue affects users of NFS file systems only.

16.2.4.1 Directories Remain After Exiting dbisql

This issue affects users of NFS file systems only.

Possible Cause

The IQTMP16 environment variable does not point to a local directory. Each client connection creates several directories and files in a temporary directory. SAP IQ deletes these files when the connection ends. If IQTMP16 does not point to a local directory, it cannot find the `.nfs*` files that NFS creates.

Action

Set IQTMP16 to a local directory and restart the server.

16.2.5 Resource Issues

Resource issues may include insufficient disk space, insufficient number of threads, thread stack overflow, and unused system resources.

In this section:

Insufficient Disk Space [page 426]
The SAP IQ server does not wait for additional space on an out-of-dbspace condition, but rolls back either the entire transaction or rolls back to a savepoint.

Monitoring Disk Space Usage [page 430]
You can use an event handler to monitor disk space usage and notify you when available space is running low.

Insufficient Threads [page 432]
The required number of server threads may not be available for your query.

Stack Overflow [page 432]
You may experience problems if the thread stack overflows.

Unused Semaphores and Shared Memory Left After Abnormal Exit [page 433]
Abnormal exits may leave unused semaphores and shared memory.

Insufficient Buffers [page 434]
If the resource manager determines that there is not enough cache to complete an operation, the operation is not started, and `Insufficient buffers` is returned.

Emergency Buffer Pool [page 434]
Operations on privileged objects, such as check database operations, may cause server failure with the message `Emergency buffer pool has no available clean buffers.`

16.2.5.1 Insufficient Disk Space

The SAP IQ server does not wait for additional space on an out-of-dbspace condition, but rolls back either the entire transaction or rolls back to a savepoint.

Context

If there is not enough temporary or main dbspace available for a buffer or dbspace allocation request, the statement making the request rolls back.

At this point, the DBA can add more space to a dbspace using the `ALTER DBSPACE` or the `ALTER FILE` command. (You may choose to add files instead of dbspaces. A single dbspace can have multiple dbfiles.)

> ⚠ Caution
>
> If SAP IQ holds certain system locks or is performing a checkpoint when you run out of disk space, you may not be able to add disk space. It is important for you to recognize when you are low on disk space, and to add a new dbspace before you run out of space.

Procedure

- Check recent messages in the SAP IQ message log (`dbname.iqmsg`). An `out of space` message indicates that you must add another dbspace. The message in the SAP IQ message file indicates which dbspace has run out of space. If the problem occurs while you are inserting data, you probably need more room in the IQ main store. If the problem occurs during queries with large sort-merge joins, you probably need more room in the IQ temporary store.text

  Check the SAP IQ message log for the following messages:
  - If a buffer or dbspace allocation request fails because there is no space in the dbspace, this error message, where %2 is the name of the dbspace, is logged in the `dbname.iqmsg` message fiel:

    ```
    You have run out of space in %2 DBSpace. %1
    ```

    ```
    [EMSG_OUT_OF_DBSPACE: SQL Code -1009170L,
    SQL State QSB66, SAP Error Code 20223]
    ```
  - If the entire transaction is rolled back on an out-of-dbspace condition, you see the following, where %1 is the error that caused the transaction to roll back, when encountered by the server during a critical operation:

    ```
    %1 -- Transaction rolled back"
    ```

    ```
    [IQ_TRANSACTION_ROLLBACK: SQL Code -1285L,
    SQL State 40W09, SAP Error Code 2973]
    ```
  - If a buffer allocation request finds a dirty buffer, but the buffer manager cannot flush the buffer due to an out-of-space condition, you see this message, where %2 is the particular buffer cache throwing the exception, and the current statement rolls back:

    ```
    %2: All buffer cache pages are in use, ask your
    DBA to increase the size of the buffer cache. %1
    ```

    ```
    [EMSG_BUFMAN_ALLSLOTSLOCKED: SQL Code -1009031L,
    SQL State QSA31, SAP Error Code 20052]
    ```
- Try to connect to the database from a new connection. If this works, the database server is running, even though the query is waiting. Run `sp_iqstatus` to get more information.
- If you cannot connect to the database, check if SAP IQ is in an unusable state by monitoring the CPU usage for that processor. If the CPU usage does not change over a small time interval, then SAP IQ is probably not operational. If the CPU usage does change, SAP IQ is operational.
- Check the `sp_iqstatus` output for:

  ```
  Main IQ Blocks Used:,10188 of 12288,
  82%, Max Block#: 134840
  ```

  ```
  Temporary IQ Blocks Used:,163 of 6144,
  2%, Max Block#: 97
  ```

  If the percentage of blocks used is in the nineties, add more disk space with the `CREATE DBSPACE` command. In this example, 82% of the Main IQ Blocks and 2% of the Temporary IQ Blocks are used, indicating that more space will soon be needed in the IQ main store.

- If out-of-space conditions occur, or `sp_iqstatus` shows a high percentage of main blocks in use on a multiplex server, run `sp_iqversionuse` to find out which versions are being used and the amount of space that can be recovered by releasing versions.

In this section:

Running out of Space During Checkpointing [page 428]
Start in forced recovery mode and add space as soon as possible.

Effect of Checkpoints on Out-of-Disk Space Conditions [page 428]
If SAP IQ has already run out of space when a checkpoint is requested, the `checkpoint` command fails with an error.

Adding Space If You Cannot Connect to a Server [page 429]
If you run out of space during an operation and cannot add space because you cannot connect to the server, add space using the `CREATE DBSPACE` command.

Managing Dbspace Size [page 429]
Growth of catalog files is normal and varies depending on application and catalog content. The size of the `.db` file does not affect performance, and free pages within the `.db` file are reused as necessary.

Adding the Wrong Type of Space [page 430]
If the temporary dbspace runs out of space and you omit the `TEMPORARY` keyword from the `CREATE DBSPACE` command, you cannot create a temporary dbspace.

Fragmentation [page 430]
SAP IQ provides control over fragmentation by taking advantage of even the smallest unused spaces.

Freeing Space [page 430]
When a connection is out of space, you cannot free space by dropping tables or indexes in another connection; the out-of-space transaction sees those objects in its snapshot version.

Reserving Space for the Future [page 430]
SAP IQ automatically reserves the minimum of 200 MB and 50 percent of the size of the last dbspace.

16.2.5.1.1 Running out of Space During Checkpointing

Start in forced recovery mode and add space as soon as possible.

You must add a dbspace before any new checkpoints can succeed.

16.2.5.1.2 Effect of Checkpoints on Out-of-Disk Space Conditions

If SAP IQ has already run out of space when a checkpoint is requested, the `checkpoint` command fails with an error.

```
You have run out of space during the CHECKPOINT operation.
```

```
[EMSG_IQSTORE_OUTOFSPACE_CHECKPOINT:'QSB33', 1009133].
```

You must add a dbspace before any new checkpoints can succeed.

16.2.5.1.3 Adding Space If You Cannot Connect to a Server

If you run out of space during an operation and cannot add space because you cannot connect to the server, add space using the `CREATE DBSPACE` command.

Procedure

1. Shut down the server using any of these methods:
   - On any platform, run `dbstop`.
   - On Windows, click the correct server icon on the Windows task bar to display the SAP IQ window, then click the **Shutdown** button.
   - On UNIX, run `stop_iq` or type `q` in the window where the server was started.
2. Restart the engine with the `start_iq` command.
3. Connect to the database.
4. Use the `CREATE DBSPACE` command to add space.
5. Re-run the operation that originally failed due to insufficient space.

16.2.5.1.4 Managing Dbspace Size

Growth of catalog files is normal and varies depending on application and catalog content. The size of the `.db` file does not affect performance, and free pages within the `.db` file are reused as necessary.

To minimize catalog file growth:

- Avoid using `IN SYSTEM` on `CREATE TABLE` statements.
- Issue `COMMIT` statements after running system stored procedures.
- Issue `COMMIT` statements after long-running transactions.

If the catalog store cannot extend one of its files (`.tmp`, `.db`, or `.iqmsg`), SAP IQ returns `A dbspace has reached its maximum file size`. To prevent this problem:

- Periodically monitor space usage.
- Verify that there are no operating system file size limits (such as Sun Solaris `ulimit`) where the `.tmp`, `.db`, or `.iqmsg` files are located. The `.db` and `.tmp` files are typically in the main SAP IQ database directory. The `.tmp` file is located under `$IQTMP16/<servername>/tmp` (for UNIX) or `%IQTMP16%\<servername>\tmp` (for Windows). If $IQTMP16 or %IQTMP16% is not set, the `.tmp` file is found under `/tmp/.SQLAnywhere/<servername>/tmp` (for UNIX) or `\tmp\.SQLAnywhere\<servername>\tmp` (for Windows).

16.2.5.1.5 Adding the Wrong Type of Space

If the temporary dbspace runs out of space and you omit the `TEMPORARY` keyword from the `CREATE DBSPACE` command, you cannot create a temporary dbspace.

Instead, add the file in the existing temporary dbspace as IQ_SYSTEM_TEMP.

16.2.5.1.6 Fragmentation

SAP IQ provides control over fragmentation by taking advantage of even the smallest unused spaces.

However, fragmentation can still occur. If your database runs out of space, even though `Mem Usage` listed by `sp_iqstatus` or the `.iqmsg` file shows that the `Main IQ Blocks Used` value is less than 100%, it usually indicates that your database is fragmented.

16.2.5.1.7 Freeing Space

When a connection is out of space, you cannot free space by dropping tables or indexes in another connection; the out-of-space transaction sees those objects in its snapshot version.

16.2.5.1.8 Reserving Space for the Future

SAP IQ automatically reserves the minimum of 200 MB and 50 percent of the size of the last dbspace.

To ensure that you have enough room to add new dbspaces if you run out of space in the future, set the database options `MAIN_RESERVED_DBSPACE_MB` and `TEMP_RESERVED_DBSPACE_MB` to values that are large enough to handle running out of space during a `COMMIT` or `CHECKPOINT`.

16.2.5.2 Monitoring Disk Space Usage

You can use an event handler to monitor disk space usage and notify you when available space is running low.

The first example in this section is especially useful for monitoring space during loads. You can enable the event handler before you start the load, and disable it after the load completes.

You can modify this sample event handler code to perform other types of monitoring.

```
-- This event handler sends email to the database
-- administrator whenever the IQ main DBSpace is more than
-- 95 percent full.
-- This event handler runs every minute. The event handler uses
-- sp_iqspaceused to sample the space usage. If the space is
-- more than 95 percent full, a file that contains the date and
```

```
-- time is created in the directory where iqsrv16 is
-- running. The file contents are then mailed to the database
-- administrator and the file is removed.
-- This event can be enabled before a load and be used
-- to monitor disk space usage during loading. The event can
-- then be disabled after the load.
create event out_of_space
schedule
start time '1:00AM' every 1 minutes
handler
begin
declare mt unsigned bigint;
declare mu unsigned bigint;
declare tt unsigned bigint;
declare tu unsigned bigint;
call sp_iqspaceused(mt, mu, tt, tu);
if mu*100/mt  > 95  then
  call xp_cmdshell('date > ./temp_m_file');
  call xp_cmdshell('mailx -s add_main_dbspace iqdba@iqdemo.com
    < ./temp_m_file');
  call xp_cmdshell('/bin/rm -rf ./temp_m_file');
end if;
if tu*100/tt  > 95  then
  call xp_cmdshell('date > ./temp_file');
  call xp_cmdshell('mailx -s add_temp_dbspace iqdba@iqdemo.com
    < ./temp_file');
  call xp_cmdshell('/bin/rm -rf ./temp_file');
end if;
end
```

The following code creates a timer-based event that monitors space usage to help avoid unexpected rollbacks, which may occur in out-of-space situations on operations without privileges. The `DBSpaceLogger` event is created in the sample `iqdemo` database.

```
CREATE EVENT DBSpaceLogger
SCHEDULE START TIME '00:00:01' EVERY 300 SECONDS
HANDLER
BEGIN
DECLARE DBSpaceName VARCHAR(128);
DECLARE Usage SMALLINT;
DECLARE cursor_1 CURSOR FOR
SELECT DBSpaceName, Usage
FROM sp_iqdbspace()
WHERE Usage > 0
ORDER BY Usage
FOR READ ONLY;
OPEN cursor_1;
idx1: LOOP
FETCH cursor_1 INTO DBSpaceName, Usage;
IF SQLCODE <> 0 THEN LEAVE idx1 END IF;
IF Usage >= 70 AND Usage < 80 THEN
call dbo.sp_iqlogtoiqmsg('Information: DBSpace' +
DBSpaceName + '''s usage is more than 70%');
ELSEIF Usage >= 80 AND Usage < 90 THEN
call dbo.sp_iqlogtoiqmsg('Warning: DBSpace ' +
DBSpaceName + '''s usage is more than 80%');
ELSEIF Usage >= 90 AND Usage < 100 THEN
call dbo.sp_iqlogtoiqmsg('Critical Warning: DBSpace
' + DBSpaceName + '''s usage is more than 90%');
END IF;
END LOOP;
CLOSE cursor_1;
END;
```

16.2.5.3 Insufficient Threads

The required number of server threads may not be available for your query.

Possible Cause

A client message similar to `Not enough server threads available for this query [-1010011] ['QXA11']` indicates that the query requires additional kernel threads for the IQ store.

Actions

- Wait for another query to finish and release the threads it is using. Then resubmit your query.
- Run `sp_iqconnection`. The column `IQThreads` contains the number of IQ threads currently assigned to the connection. This column can help you determine which connections are using the most resources. Some threads may be assigned but idle.
- If the condition persists, you may need to restart the server and specify additional IQ threads. Use the `-iqmt` server startup switch to increase the number of processing threads that SAP IQ can use.
  The default is 60 threads per CPU for the first 4 CPUs, and 50 threads per CPU for the remainder, with 3 more for system use, plus threads needed for database connections and background tasks. For example, on a system with 12 CPUs and 10 connections: `60*4 + 50*(numCPUs - 4) + numConnections + 6 = 656`. The minimum value is `numConnections + 3`. The total number of server threads cannot exceed 10000.
- If the server runs out of threads, or if sufficient threads are not available to a connection during a restore, you may see the error `Ran out of threads. Start up server with more threads.` (SQLCODE -1012024). The `RESTORE` command attempts to allocate a "team" of threads for the restore operation. SAP IQ attempts to allocate at least one thread per backup device, plus two threads per CPU, plus one thread to the team. Make sure you have allocated enough threads on both a per-connection and per-team basis, as well as to the server. Use the `MAX_IQ_THREADS_PER_CONNECTION` and `MAX_IQ_THREADS_PER_TEAM` database options.

16.2.5.4 Stack Overflow

You may experience problems if the thread stack overflows.

`AbortIfEndofStack` in the stack trace file (`stktrc-YYYYMMDD-HHNNSS_#.iq`), indicates that the thread stack has overflowed.

Possible Causes

- To avoid this problem, restart SAP IQ with the server parameter `-iqtss` set to 300 on 32-bit operating systems, or 500 on 64-bit operating systems. The server startup switch `-iqtss` specifies thread stack size in KB. If this is inadequate, raise the value of `-iqtss` by 72 until the problem is solved.
- If possible, identify the command that caused the error and forward it to Technical Support.

16.2.5.5 Unused Semaphores and Shared Memory Left After Abnormal Exit

Abnormal exits may leave unused semaphores and shared memory.

AIX, HP-UX, and Linux platforms use semaphores for communication between clients and servers on the same computer. Each client allocates one semaphore, as does each server. A client signals the server's semaphore when it has placed a packet for the server to read, and vice versa. The number of semaphores needed for a given system depends on how many local client applications connect via shared memory to the local server. If a client needs to allocate multiple semaphores for multiplex connections to one or more servers, it attempts to allocate all semaphores in the semaphore group.

Possible Causes

Killing processes on UNIX systems may result in semaphores or shared memory being left behind instead of being cleaned up automatically. To eliminate unneeded semaphores, periodically run the UNIX `ipcs` command to check the status of semaphores and shared memory.

The `ipcs -a` command lists the ID numbers, owners, and create times of semaphores and shared memory segments. When all SAP IQ instances are started by the same user (this is recommended), you can search the OWNER column for that user name. Identify shared memory segments and semaphores that are not being used.

Action

After verifying with the owner that these shared memory segments and semaphores are not in use, run the UNIX `ipcrm` command to remove them. Use the `-m` parameter to specify the memory segment ID and the `-s` command to specify the semaphore ID number, in the following format:

```
ipcrm -m mid1 -m mid2 ... -s sid1 -s sid2 ...
```

For example:

```
% ipcrm -m 40965 -s 5130 -s36682
```

16.2.5.6 Insufficient Buffers

If the resource manager determines that there is not enough cache to complete an operation, the operation is not started, and `Insufficient buffers` is returned.

Use one of these suggestions to resolve the issue:

- Increase the buffer cache size and re-run the operation.
- Reschedule the operation to a time when the server is not as busy and more buffer cache might be available.
- In a multiplex environment, move the workload to another node.

16.2.5.7 Emergency Buffer Pool

Operations on privileged objects, such as check database operations, may cause server failure with the message `Emergency buffer pool has no available clean buffers.`

The emergency buffer pool prevents lack of memory buffers from blocking privileged objects such as the global freelist, TLV log, check database operations, checkpoint log, and blockmap metadata pages.

Action

If you encounter this error, double the default value of the N_EMERG_BUFFERS option and restart the server. The value default is 20.

For example:

```
SET OPTION PUBLIC.N_EMERG_BUFFERS = 40
```

If you start the server with the -iqpartition option, the emergency buffer pool is subdivided. For example, if the server in the previous example started with -iqpartition 32, each of 32 partitions is allocated 40 buffers.

16.2.6 Processing Issues

Processing issues may be related to loads, queries, indexes, and table access.

In this section:

Too Many Indexes on Table [page 435]
Issues may occur when a table has too many indexes.

Unexpectedly Long Loads or Queries [page 435]
Long loads or queries may cause issues.

Cannot Write to a Locked Table [page 436]
Locked tables may cause issues.

Managing Write Lock Contention on a Table [page 437]
: High contention for write locks on a table used by multiple users can impact processing, if most of the transactions can obtain the lock.

Checkpoint Hints [page 438]
: The default values for checkpoint time and recovery time are sufficient and in most cases do not need to be changed.

Dbfile Creation Slower Than Expected [page 438]
: Dbfile creation takes longer than expected.

16.2.6.1 Too Many Indexes on Table

Issues may occur when a table has too many indexes.

Possible Cause

A Microsoft Access user is trying to link to a table that has more than 32 indexes.

Action

Create a view that selects all the columns in the table, and link to the view instead of the base table.

16.2.6.2 Unexpectedly Long Loads or Queries

Long loads or queries may cause issues.

Possible Causes

- IQ buffer cache is too large, so the operating system is thrashing.
- IQ buffer cache is too small, so SAP IQ is thrashing because it cannot fit enough of the query data into the cache.
- You attempted to set IQ buffer cache sizes so that total memory requirements on your system exceed total system memory. Consequently, buffer caches have been therefore automatically reduced to their default sizes.
- User-defined functions or cross-database joins requiring CIS intervention.
- Missing HG index on columns used in the `WHERE` clause and `GROUP BY` clause.

Action

Monitor paging to determine if thrashing is a problem.

- To monitor IQ paging, run the IQ buffer cache monitor.
- To monitor operating system paging, use the UNIX `vmstat` utility or other platform-specific tools, or the Windows Performance Monitor.

Reset your buffer sizes as needed.

You can also limit the amount of thrashing during a query execution that involves hash algorithms. Adjusting the `HASH_THRASHING_PERCENT` database option controls the percentage of hard disk I/Os allowed before the statement is rolled back and an error is returned.

The default value of `HASH_THRASHING_PERCENT` is 10%. Increasing the value permits more paging to disk before a rollback, and decreasing the value permits less paging before a rollback.

Queries involving hash algorithms that executed in earlier versions of SAP IQ may now be rolled back when the default `HASH_THRASHING_PERCENT` limit is reached, and you may see either of these messages:

- `Hash insert thrashing detected.`
- `Hash find thrashing detected.` (SQLState QFA43, SQLCode -1001047)

To provide the query with the resources required for execution, perform one or more of these actions:

- Relax the paging restriction by increasing the value of `HASH_THRASHING_PERCENT`.
- Increase the size of the temporary cache (DBA only). Increasing the size of the temporary cache reduces the size of the main buffer cache.
- Attempt to identify and alleviate why SAP IQ is incorrectly estimating one or more hash sizes for this statement.
- Decrease the value of the database option `HASH_PINNABLE_CACHE_PERCENT`.

To identify possible problems with a query, generate a query plan by running the query with the temporary database options QUERY_PLAN = 'ON' and QUERY _DETAIL = 'ON', then examine the estimates in the query plan. The option QUERY_PLAN_AFTER_RUN = 'ON' provides additional information, as the query plan is printed after the query has finished running. The generated query plan is in the message log file.

16.2.6.3 Cannot Write to a Locked Table

Locked tables may cause issues.

Possible Causes

The following error message is reported when writing to an object to which another user already has write access.

```
Cannot open the requested object for write in the current transaction (TxnID1).
Another user has write access in transaction TxnID2.
```

Action

Use the `sp_iqlocks` stored procedure to identify users who are blocking other users from writing to a table. This procedure displays information about locks currently held in the database, including the connection and user ID that holds the lock, the table on which the lock is held, the type of lock, and a name to identify the lock.

The error message also includes the transaction ID of the user who is attempting to write (TxnID1) and the transaction ID of the user who is currently writing (TxnID2). For more detailed information about the transaction that has locked the table, run the `sp_iqtransaction` stored procedure.

16.2.6.4 Managing Write Lock Contention on a Table

High contention for write locks on a table used by multiple users can impact processing, if most of the transactions can obtain the lock.

This sample stored procedure shows one method of managing contention for a write lock on a table. This procedure does not eliminate the write lock contention on the table, but does manage the contention, so that transactions can obtain the write lock.

The sample stored procedure code manages lock contention on a table named `dbo.event`, which records events. The procedure returns the `event_id` to the caller. This table is in high contention for write locks. The stored procedure `dbo.log_event` records information in the table `dbo.event`. If an access error occurs, the error is captured, the hopeful writer sleeps for five seconds, and then attempts to write to the table again. The five second retry interval is usually long enough for the contention to be resolved, so the write lock on the `dbo.event` table is available.

You can modify this code to perform other similar tasks.

```
if exists (select 1
           from   sys.sysprocedure a
           join   sys.sysuserperm b on a.creator = b.user_id
           where  a.proc_name = 'log_event' and b.user_name = 'dbo') then
   drop procedure dbo.log_event;
end if;
create procedure dbo.log_event(in @event varchar(255))
on exception resume
begin
    declare @event_id   bigint;
    declare @res        char(5);
    set @event_id=0;
    loop1: loop
        commit work;
        select  max(event_id)+1
            into    @event_id
            from    dbo.event;
        insert  dbo.event
            values (@event_id,@event,current timestamp,null,null);
        set @res=sqlstate;
        if @res = ' ' or(@res <> 'QDA29' and @res <> 'QDA11') then
            leave loop1
        end if;
        call dbo.sleep(5);
    end loop loop1;
    commit work;
    return @event_id
end
```

To prevent a critical update operation from failing, you may reserve write locks on all required tables in advance. For example, the following example reserves write locks on the tables `SalesOrders`, `Customers`, and `SalesOrderItems`, which are required for a hypothetical update:

```
BEGIN
WHILE TRUE LOOP
    LOCK TABLE SalesOrders, SalesOrderItems, Customers IN WRITE MODE WAIT
'30:00:00';
    If SQLCODE indicates that lock could not be acquired
    then
        SET status_msg = 'lock for required tables
        not yet acquired – retrying';
        Message to client status_msg;
    ELSE
        BREAK;
    ENDIF;
END LOOP; // Locks on SalesOrders, SalesOrderItems, Customers are acquired
Update table SalesOrders …;
INSERT INTO SalesOrderItems …;
LOAD INTO Customers …;
COMMIT;
END;
```

16.2.6.5 Checkpoint Hints

The default values for checkpoint time and recovery time are sufficient and in most cases do not need to be changed.

The time between checkpoints defaults to 60 minutes.

You can adjust the time between checkpoints when you start your server by changing the `-gc` and `-gr` options in the `start_iq` command or in the `dbname.cfg` configuration file. The `-gc` switch specifies the number of minutes for the checkpoint timeout period. The `-gr` switch specifies the number of minutes for the maximum recovery time. The database engine uses both switches to calculate the checkpoint time.

For details on `start_iq` database options, see the *SAP IQ Utility Reference*.

16.2.6.6 Dbfile Creation Slower Than Expected

Dbfile creation takes longer than expected.

Possible Cause

A slow underlying file system slows down dbfile creation.

Action

If you suspect a slow file system, check the `.iqmsg` file for messages like these:

```
I. 03/30 11:31:05. 0000013261 139619391600384 Info: posix_fallocate fd=627
sample time 0.220000 sec.
I. 03/30 11:31:05. 0000013261 139619391600384 Info: posix_fallocate fd=627
second sample time 0.000000 sec.
I. 03/30 11:31:05. 0000013261 139619391600384 Info: posix_fallocate fd=627
sample time 0.000000 sec.
I. 03/30 11:31:05. 0000013261 139619391600384 Info: using posix_fallocate
fd=627.  Sampling threshold was 0.050000 sec.
```

Convert the current value of FILE_PREALLOCATE_SAMPLING_THRESHOLD to seconds and compare it to the values of sample time and second sample time. If the lower of sample time and second sample time exceeds the value of FILE_PREALLOCATE_SAMPLING_THRESHOLD, set FILE_PREALLOCATE_SAMPLING_THRESHOLD to a value greater than the smaller of the sample time and second sample time values.

16.2.7 Performance Issues

Certain settings may impact performance.

In this section:

Slow Performance on a Multi-CPU or Hyperthreaded Machine [page 439]
Multi-CPU or hyperthreaded machines may experience issues.

16.2.7.1 Slow Performance on a Multi-CPU or Hyperthreaded Machine

Multi-CPU or hyperthreaded machines may experience issues.

Possible Cause

SAP IQ runs most efficiently when it knows how many physical CPUs are available to it. On a machine with hyperthreads turned on, or where SAP IQ cannot access all of the available CPUs, SAP IQ creates too many threads and run less efficiently than it should.

Action

Start the server with `-iqnumbercpus` set to the number of CPUs available to SAP IQ, overriding the physical number of CPUs.

For details on `start_iq` database options, see the *SAP IQ Utility Reference*.

16.3 Troubleshooting Network Communications

Network software involves several different components, increasing the likelihood of issues requiring troubleshooting.

The primary source of assistance in network troubleshooting is the documentation and technical support for your network communications software, as provided by your network communications software vendor. However, you can follow best practices and use diagnostic tools to obtain information on various conditions.

In this section:

Using Compatible Protocols [page 441]
: If you have more than one protocol stack installed on the client or server computer, ensure that the client and the database server are using the same protocol.

Using Current Drivers [page 441]
: Ensure that you have the latest version of the NDIS or ODI driver for your network adapter, as appropriate.

Powering Down Your Computer Between Restarts [page 441]
: Some network adapter boards do not reset cleanly when you restart the computer. When you are troubleshooting, turn the computer off, wait a few seconds, and then turn it back on.

Diagnosing the Protocol Stack Layer by Layer [page 441]
: If you are having problems getting your client application to communicate with a database server, ensure that the client and the database server are using compatible protocol stacks.

Testing a TCP/IP Protocol Stack [page 442]
: If you are running under TCP/IP, there are several applications you can use to test the compatibility of the client computer and server computer TCP/IP protocol stack.

Diagnosing Wiring Problems [page 443]
: Faulty network wiring or connectors can cause problems that are difficult to isolate.

Checking Common Network Communications Problems [page 443]
: Familiarize yourself with common network communications problems and their solutions.

16.3.1 Using Compatible Protocols

If you have more than one protocol stack installed on the client or server computer, ensure that the client and the database server are using the same protocol.

The `-x` command line switch for the server selects a list of protocols for the server to use, and the CommLinks connection parameter does the same for the client application.

You can use these options to ensure that each application is using the same protocol.

By default, both the database server and client library use all available protocol stacks. The server supports client requests on any active protocol, and the client searches for a server on all active protocols.

For more information about the `start_iq` database startup utility `-x` switch, see the *SAP IQ Utility Reference*.

16.3.2 Using Current Drivers

Ensure that you have the latest version of the NDIS or ODI driver for your network adapter, as appropriate.

You should be able to obtain current network adapter drivers from the manufacturer or supplier of the adapter card.

Network adapter manufacturers and suppliers make the latest versions of drivers for their cards available. Most card manufacturers have a Web site from which you can download the latest versions of NDIS and ODI drivers.

You may also be able to obtain a current network adapter driver from the provider of your networking software.

When you download Novell client software, ODI drivers for some network adapters are included, in addition to the Novell software that is used for all network adapters.

16.3.3 Powering Down Your Computer Between Restarts

Some network adapter boards do not reset cleanly when you restart the computer. When you are troubleshooting, turn the computer off, wait a few seconds, and then turn it back on.

16.3.4 Diagnosing the Protocol Stack Layer by Layer

If you are having problems getting your client application to communicate with a database server, ensure that the client and the database server are using compatible protocol stacks.

One way to isolate network communication problems is to work up the protocol stack, testing whether each level of communication is working properly.

If you can connect to the server computer, the data link layer is working, regardless of whether the connection is made using the same higher-layer protocols you will be using for SAP IQ.

For example, try to connect to a disk drive on the computer running the database server from the computer running the client application.

Once you have verified that the data link layer is working, next verify other applications using the same network and transport layers as SAP IQ are working.

16.3.5 Testing a TCP/IP Protocol Stack

If you are running under TCP/IP, there are several applications you can use to test the compatibility of the client computer and server computer TCP/IP protocol stack.

In this section:

Using Ping to Test the IP Layer [page 442]
Each IP layer has an associated address—a four-integer period-separated number (such as 191.72.109.12). `ping` takes as an argument an IP address and attempts to send a single packet to the named IP protocol stack.

Using Telnet to Test the TCP/IP Stack [page 443]
To further test the TCP/IP stack, start a server application on one computer, and a client program on the other computer, and test whether they can communicate properly.

16.3.5.1 Using Ping to Test the IP Layer

Each IP layer has an associated address—a four-integer period-separated number (such as 191.72.109.12). `ping` takes as an argument an IP address and attempts to send a single packet to the named IP protocol stack.

First, determine if your own protocol stack is operating correctly by "pinging" your own computer. For example, if your IP address is 191.72.109.12, enter this command at the command prompt:

```
ping 191.72.109.12
```

Wait to see if the packets are routed. If they are, you see output similar to:

```
c:> ping 191.72.109.12
Pinging 191.72.109.12 with 32 bytes of data:
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
...
```

If the ping works, it indicates that the computer can route packets to itself. This is reasonable assurance that the IP layer is set up correctly. Ask someone else running TCP/IP for his or her IP address, and try pinging that computer.

Before proceeding with additional diagnostics, ensure that you can ping the computer running the database server from the client computer.

16.3.5.2 Using Telnet to Test the TCP/IP Stack

To further test the TCP/IP stack, start a server application on one computer, and a client program on the other computer, and test whether they can communicate properly.

There are several applications commonly provided with TCP/IP implementations that can be used for this purpose. To use the `telnet` command to test the TCP/IP stack:

1. Start a Telnet server process (or daemon) on one machine. Check your TCP/IP software documentation for instructions. For a typical command line Telnet program, enter this at the command prompt:

   ```
   telnetd
   ```

2. Start the Telnet client process on the other machine, and see if you get a connection. Again, check your TCP/IP software documentation for instructions. Typically, enter an instruction similar to the following, where `<server_name>` is the name or IP address of the computer running the Telnet server process:

   ```
   telnet <server_name>
   ```

Establishing a Telnet connection between these two machines indicates that the protocol stack is stable and the client and server should be able to communicate using the TCP/IP link. If you cannot establish a Telnet connection, there is a problem. Before proceeding with additional diagnostics, ensure that your TCP/IP protocol stack is working correctly.

16.3.6 Diagnosing Wiring Problems

Faulty network wiring or connectors can cause problems that are difficult to isolate.

Try re-creating problems on a similar machine with the same configuration. If a problem occurs on only one machine, it may indicate a wiring or hardware problem.

For information on detecting wiring problems under NetWare, see your Novell NetWare manuals. The Novell LANalyzer program is useful for diagnosing wiring problems with Ethernet or TokenRing networks. Your NetWare authorized reseller can also supply you with the name of a Certified NetWare Engineer who can help diagnose and solve wiring problems.

16.3.7 Checking Common Network Communications Problems

Familiarize yourself with common network communications problems and their solutions.

In this section:

"Unable to start — server not found" Message [page 444]
The message `Unable to start — server not found` when trying to start the client, indicates that the client cannot find the database server on the network.

"Unable to initialize any communication links" Message [page 444]

The message `Unable to initialize any communication links`, indicates that no link can be established.

16.3.7.1 "Unable to start — server not found" Message

The message `Unable to start — server not found` when trying to start the client, indicates that the client cannot find the database server on the network.

- The network configuration parameters of your network driver on the client machine may be different from those on the server machine. For example, two Ethernet adapter cards should use a common frame type. For Novell NetWare, the frame type is specified in the `net.cfg` file. In Windows, look for the frame type setting in the Control Panel Network Settings.
- Under the TCP/IP protocol, clients search for database servers by broadcasting a request. Such broadcasts typically do not pass through gateways, so any database server on a machine in another (sub)network, is not found. In this case, you must supply the host name of the machine on which the server is running using the `-x` server startup command line option. This is required to connect to NetWare servers over TCP.
- Your network drivers or wiring is not installed properly.
- The network configuration parameters of your network driver may be incompatible with SAP IQ multiuser support.

16.3.7.2 "Unable to initialize any communication links" Message

The message `Unable to initialize any communication links`, indicates that no link can be established.

The probable cause is that your network drivers have not been installed. The server and the client try to start communication links using all available protocols, unless you have specified otherwise using the `-x` server startup option. Check your network documentation to find out how to install the driver you need to use.

16.4 Diagnostic Tools

Several tools help you diagnose various conditions.

In this section:

Restoring to a New Temporary File Topology [page 445]
If temporary dbfiles cannot be opened or are damaged, you can restore the database to a different temporary file topology.

The sp_iqstatus Stored Procedure [page 446]
The `sp_iqstatus` stored procedure provides a variety of SAP IQ status information.

Interpreting Notification Messages [page 448]
By default, SAP IQ displays information about your database during insert and load operations in the IQ message log (`.iqmsg` file).

The sp_iqcheckdb Stored Procedure [page 453]
If you suspect problems in your database, try running the stored procedure `sp_iqcheckdb`.

Checking Database and Server Startup Option Values [page 453]
When diagnosing server startup, resource, or processing issues, you may need to check the current values of database options and server startup options.

Finding the Currently Executing Statement [page 454]
When diagnosing a problem, you may want to know what statement was executing when the problem occurred.

Logging Server Requests [page 454]
To isolate some types of problems, especially problems with queries, log server requests.

Connection for Collecting Diagnostic Information [page 458]
The database option `DEDICATED_TASK` lets the DBA dedicate a request handling task to handle requests from a single connection.

Diagnosing Communications Issues [page 459]
If your server returns a communication error on startup, you may want to set the `-z` command line option when you start the server.

16.4.1 Restoring to a New Temporary File Topology

If temporary dbfiles cannot be opened or are damaged, you can restore the database to a different temporary file topology.

Procedure

1. Start the utility server so it ignores all temporary IQ file definitions in the backed up database during the restore:

```
start_iq -n utility_startup_svr -c 32m
-x 'tcpip{port=1234}' -iqnotemp
```

2. Restore the database:

```
RESTORE DATABASE 'iqdemo'
FROM '/system1/IQ16/IQ-16_1/demo/backup/iqmain'
```

3. Restart the restored database using the `-iqnotemp` flag.
4. Drop all the files in IQ_SYSTEM_TEMP:

```
ALTER DBSPACE IQ_SYSTEM_TEMP DROP FILE ALL
```

5. Restart the server without the `-iqnotemp` flag.

6. Add new temporary dbfiles to IQ_SYSTEM_TEMP

16.4.2 The sp_iqstatus Stored Procedure

The `sp_iqstatus` stored procedure provides a variety of SAP IQ status information.

> **i Note**
>
> The following example shows output from the `iqdemo` sample database. The sample user dbspace `iq_main` may not be present in your own user-created databases.

The following output is from the `sp_iqstatus` stored procedure:

SAP IQ (TM)	Copyright (c) 1992-2016 by SAP AG or an SAP affiliate company. All rights reserved.
Version:	16.1.010.844/10172/P/Mainline/Linux64 - x86_64 - 2.6.18-194/64bit/ 2016-12-22 02:31:33
Time Now:	2017-01-18 20:14:57.664
Build Time:	2016-12-22 02:31:33
File Format:	23 on 03/18/1999
Server mode:	IQ Server
Catalog Format:	2
Stored Procedure Revision:	1
Page Size:	131072/8192blksz/16bpp
Number of Main DB Files:	2
Main Store Out Of Space:	N
Number of Cache Dbspace Files:	0
Number of Shared Temp DB Files:	0
Shared Temp Store Out Of Space:	N
Number of Local Temp DB Files:	1
Local Temp Store Out Of Space:	N
DB Blocks: 1-12800	IQ_SYSTEM_MAIN
DB Blocks: 1045440-1058239	iq_main
Local Temp Blocks: 1-3200	IQ_SYSTEM_TEMP
Create Time:	2017-01-18 19:09:30.231
Update Time:	2017-01-18 19:09:34.000
Main IQ Buffers:	509
Temporary IQ Buffers:	509
Main IQ Blocks Used:	5944 of 19200

Cache Dbspace IQ Blocks Used:	0 of 0
Shared Temporary IQ Blocks Used:	0 of 0
Local Temporary IQ Blocks Used:	65 of 1600
Main Reserved Blocks Available:	6400 of 6400
Shared Temporary Reserved Blocks Available:	0 of 0
Local Temporary Reserved Blocks Available:	1600 of 1600
IQ Dynamic Memory:	Current: 288mb
IQ Heap Memory:	Current: 150mb
Main IQ Buffers:	Used: 6
Temporary IQ Buffers:	Used: 4
Main IQ I/O:	I: L200/P6 O: C0/D22/P20 D:0 C:100.0
Temporary IQ I/O:	I: L710/P0 O: C124/D134/P13 D:120 C:100.0
Other Versions:	0 = 0Mb
Active Txn Versions:	0 = C:0Mb/D:0Mb
Last Full Backup ID:	0
Last Full Backup Time:	
Last Backup ID:	0
Last Backup Type:	None
Last Backup Time:	
DB Updated:	1
Blocks in next ISF Backup:	0 Blocks: =0Mb
Blocks in next ISI Backup:	0 Blocks: =0Mb
IQ large memory space:	2048Mb
IQ large memory flexible percentage:	50
IQ large memory flexible used:	0Mb
IQ large memory inflexible percentage:	90
IQ large memory inflexible used:	0Mb
IQ large memory anti-starvation percentage:	50
DB File Encryption Status:	OFF
RLV Status:	RW
RLV memory limit (mb):	2048
RLV memory used (bytes):	0
RLV Log Buffers Allocated:	0
RLV Log Buffers Globally Free:	0
RLV Log Buffers Privately Free:	0
RLV Log Buffers In Use:	0

Key to `Main IQ I/O` and `Temporary IQ I/O` output codes:

- `I` – Input
- `L` – Logical pages read (“Finds”)
- `P` – Physical pages read
- `O` – Output
- `C` – Pages Created
- `D` – Pages Dirtied
- `P` – Physically Written
- `D` – Pages Destroyed
- `C` –Compression Ratio

Check the following information:

- The lines `Main IQ Blocks Used` and `Temporary IQ Blocks Used` tell what portion of your dbspaces is in use. If the percentage of blocks in use (the middle statistic on these lines) is in the high nineties, add a dbspace.
- The `Main IQ Blocks Used` and `Temporary IQ Blocks Used` are calculated based on the line `DB Blocks` (Total Main IQ Blocks) minus `Main Reserved Blocks Available`and the line `Temp Blocks` (Total Temp IQ Blocks) minus `Temporary Reserved Blocks Available`, since the Reserved Blocks cannot be used for user operations.
- The lines `Main IQ Buffers` and `Temporary IQ Buffers` tell you the current sizes of your main and temp buffer caches.
- `Other Versions` shows other db versions and the total space consumed. These versions will eventually be dropped when they are no longer referenced or referenceable by active transactions.
- `Active Txn Versions` shows the number of active write transactions and the amount of data they have created and destroyed. If these transactions commit, the “destroyed” data becomes an old version and eventually be dropped. If they roll back, the “created” data is freed.
- `Main Reserved Blocks Available` and `Temporary Reserved Blocks Available` show the amount of available reserved space.
- The lines `Main IQ I/O` and `Temporary IQ I/O` display I/O status in the same format as in the IQ message log.

16.4.3 Interpreting Notification Messages

By default, SAP IQ displays information about your database during insert and load operations in the IQ message log (`.iqmsg` file).

The statistics in these messages indicate when you need to perform maintenance and optimization tasks, such as adding more dbspaces. The messages also report on the progress of the load.

At the start of the insert is a description of the operation, such as:

```
In table 'tab2', the full width insert
of 2 columns will begin at record 1.
I. 02/11 13:28:14. 0000000002 Insert Started:
I. 02/11 13:28:14. 0000000002 tab2
I. 02/11 13:28:14. 0000000227 [20895]: Insert Pass 1
completed in 0 seconds.
I. 02/11 13:28:14. 0000000227 [20895]: Insert Pass 2
```

```
completed in 0 seconds.
I. 02/11 13:28:14. 0000000227 [20834]:
    1 records were inserted into 'tab2'.
```

Each time SAP IQ inserts the number of records specified in the `NOTIFY` load option, the server sends a message such as the following:

```
2010-05-27 13:03:49 0000000002
[20897]: 100000 Records, 2 Seconds
```

The first line shows how many rows SAP IQ has read so far and the number of seconds taken since the last notification message to read these additional rows. Even if SAP IQ reads the same number of rows each time, the amount of time varies depending on the data read (for example, how many data conversions are required). Reported time intervals smaller than 1 second are usually reported as "0 Secs".

In this section:

Memory Message [page 449]
The memory message displays information about memory usage of the SAP IQ server.

IQ Main Store Blocks Message [page 450]
The IQ main store blocks message displays information about use of the blocks and buffers in the IQ main store.

IQ Temporary Store Blocks Message [page 450]
The IQ temporary store blocks message displays information about use of the blocks and buffers in the IQ temporary store.

Main Buffer Cache Activity Message [page 451]
The main buffer cache activity message displays information about the IQ main store buffer cache.

Temporary Buffer Cache Message [page 452]
The temporary buffer cache activity message displays information about the IQ temporary store buffer cache.

User Name, Connection Handle, and Connection ID [page 453]
After the temporary buffer cache message, the connection handle, connection ID (SA connID), and user name are logged in the `.iqmsg` file once per database connection.

16.4.3.1 Memory Message

The memory message displays information about memory usage of the SAP IQ server.

This line in the IQ message log (`.iqmsg` file) displays memory usage information:

```
Mem: 469mb/M470
```

Memory usage message:

- Mem: # mb – current memory, in megabytes, being used by this SAP IQ server.
- M# mb – the maximum number of megabytes used by this SAP IQ server since it was started.

16.4.3.2 IQ Main Store Blocks Message

The IQ main store blocks message displays information about use of the blocks and buffers in the IQ main store.

This line in the IQ message log (`.iqmsg` file) describes the permanent IQ main store:

```
Main      Blks: U63137/6%, Buffers: U12578/L7
```

Item	Description
U#	Number of blocks in use.
#%	Percentage of database filled.
Buffers: U#	Number of buffers in use. Normally this is 100% because the buffer manager leaves buffers in memory until they are needed for some other data. In general, the buffers used and buffers locked numbers are meaningless, because SAP IQ uses buffers as aggressively and efficiently as it can. This value grows to the maximum number of buffers that fit in the main buffer cache. The number increments whenever a buffer is allocated, but decrements only when a buffer is destroyed, not when it is unlocked or flushed. Objects in the temporary cache release their buffers when they are finished, but in the main buffer cache, SAP IQ may or may not destroy the buffers because as long as a buffer is unlocked, it is available for reuse, whether it is empty, contains data, or contains destroyed data.
L#	Number of locked buffers. A locked buffer is in use and cannot be removed from the cache. SAP IQ locks buffers of some objects, such as hash objects, to keep them in memory. It locks buffers of other objects, such as sorts, depending on the workload and what it considers a fair share for that object. This number increments whenever you request a buffer. If you exceed the maximum while running a script, the command that exceeds fails and subsequent commands may complete incorrectly. Buffer locks do not use any memory. A locked buffer has a flag set in the in-memory structure and the flag exists whether or not the buffer is locked.

It is important that you recognize when the server is low on disk space and that you add a new dbspace before the server runs out of space. For an example of using an event handler to monitor disk space usage and to notify you when available space is low during a load, see *Monitoring Disk Space Usage*.

16.4.3.3 IQ Temporary Store Blocks Message

The IQ temporary store blocks message displays information about use of the blocks and buffers in the IQ temporary store.

This line in the IQ message log (`.iqmsg` file) describes the Temporary IQ Store:

```
Temporary Blks: U273/0%, Buffers: U1987/L1960
```

Item	Description
U#	Number of blocks in use.
#%	Percentage of database filled.
Buffers: U#	Number of buffers in use. Normally, this is 100% because the buffer manager leaves buffers in memory until they are needed for some other data. In general, the buffers used and buffers locked numbers are meaningless, because IQ uses buffers as aggressively and efficiently as it can. Objects in the temporary cache release their buffers when they are finished.
L#	Number of locked buffers. A locked buffer is in use and cannot be removed from the cache. IQ locks buffers of some objects, such as hash objects, to keep them in memory. It locks buffers of other objects, such as sorts, depending on the workload and what it considers a fair share for that object. This number increments when you request a buffer. If you exceed the maximum while running a script, the command that exceeds it fails and subsequent commands may complete incorrectly. Buffer locks do not use any memory. A locked buffer has a flag set in the in-memory structure and the flag exists whether or not the buffer is locked.

It is important that you recognize when the server is low on disk space and adding a new dbspace before the server runs out of space is important. For an example of using an event handler to monitor disk space usage and to notify you when available space is low during a load, see Monitoring Disk Space Usage [page 430].

16.4.3.4 Main Buffer Cache Activity Message

The main buffer cache activity message displays information about the IQ main store buffer cache.

This line in the IQ message log (`.iqmsg` file) displays information about the IQ main store buffer cache:

```
Main      I: L331224/P22 O: D25967/P7805 C:D0
```

Item	Description
Main: I: L#	Number of logical file reads.
P#	Number of physical file reads.
O: D#	Number of times a buffer was destroyed.
P#	Number of physical writes.

Item	Description
C: D#	Buffer manager data compression ratio. This is the total number of bytes eligible for compression, minus the number of bytes used after compression, divided by total number of bytes eligible for compression times 100. In other words, how much data has been compressed (what percentage it is of its uncompressed size). The larger the number, the better. Only certain data blocks are eligible for compression. Eligible blocks include indexes, (90-95% of a database) and sort sets. The sort reflects only data compression techniques used by the buffer manager. Other data compression may take place before data reaches the buffer manager, so the total data compression may be higher.

In general, assuming the buffer cache is full, you should have between 10 and 1000 logical reads per physical read. A lower value indicates excessive thrashing in the buffer manager. More than 1000 times larger can indicate that you may have allocated too much memory to your buffer cache.

16.4.3.5 Temporary Buffer Cache Message

The temporary buffer cache activity message displays information about the IQ temporary store buffer cache.

This line in the IQ message log (`.iqmsg` file) displays information about the IQ temporary store buffer cache:

```
Temporary I: L25240/P8 O: D4749/P0 C:D0
```

Item	Description
Temporary: I: L#	Number of logical file reads.
P#	Number of physical file reads.
O: D#	Number of times a buffer was destroyed.
P#	Number of physical writes.
C: D#	Buffer manager data compression ratio. This is the total number of bytes eligible for compression, minus number of bytes used after compression, divided by the total number of bytes eligible for compression, times 100. In other words, how much data has been compressed (what percentage it is of its uncompressed size). The larger the number, the better. Only certain data blocks are eligible for compression. Eligible blocks include indexes, (90-95% of a database) and sort sets. The ratio reflects only data compression techniques used by the buffer manager. Other data compression may take place before data reaches the buffer manager, so the total data compression may be higher.

In general, assuming the buffer cache is full, you should have between 10 and 1000 logical reads per physical read. A lower value indicates excessive thrashing in the buffer manager. More than 1000 times larger can indicate that you may be overallocating memory to your buffer cache.

16.4.3.6 User Name, Connection Handle, and Connection ID

After the temporary buffer cache message, the connection handle, connection ID (SA connID), and user name are logged in the `.iqmsg` file once per database connection.

These lines in the IQ message log (`.iqmsg` file) display connection information:

```
2010-05-12 09:34:42 0000000002 Txn 173
2010-05-12 09:34:42 0000000002 Connect: 1550990889. SA connID: 1. User: DBA.
```

The connection handle is the value shown by the `sa_conn_info` stored procedure.

> **i** Note
>
> To correlate connection information in the `-zr` log file with that in the `.iqmsg` file, see *Correlating connection information between the .srvlog and .iqmsg files*.

16.4.4 The sp_iqcheckdb Stored Procedure

If you suspect problems in your database, try running the stored procedure `sp_iqcheckdb`.

This procedure reads every database page from disk into memory and performs various consistency checks. However, depending on the size of your database, the check can take a long time to run.

`sp_iqdbstatistics` displays the database statistics collected by the most recent execution of `sp_iqcheckdb`.

16.4.5 Checking Database and Server Startup Option Values

When diagnosing server startup, resource, or processing issues, you may need to check the current values of database options and server startup options.

For the connected user, the `sp_iqcheckoptions` stored procedure displays a list of the current value and the default value of database options that have been changed from the default. `sp_iqcheckoptions` also lists server startup options that have been changed from the default values.

When a DBA executes `sp_iqcheckoptions` he or she sees all options that are set on a permanent basis for all roles and users, and sees temporary options set for DBA. Non-DBA users see their own temporary options. All users see non-default server startup options.

The `sp_iqcheckoptions` stored procedure requires no parameters. In Interactive SQL, execute the following:

```
sp_iqcheckoptions
```

The system table DBO.SYSOPTIONDEFAULTS contains all of the names and default values of the SAP IQ and SAP SQL Anywhere options. You can query this table, to see all option default values.

16.4.6 Finding the Currently Executing Statement

When diagnosing a problem, you may want to know what statement was executing when the problem occurred.

The `sp_iqcontext` stored procedure lists currently running statements, and identifies the user and connection that issued the statement. You can use this utility together with information provided by `sp_iqconnection`, the `.iqmsg` log, and the `-zr` server request log (`.srvlog`), as well as stack traces, to determine what was happening when a problem occurred.

To match `.iqmsg` log and the `-zr` server request log entries using connection information, correlate connection information between the `.srvlog` and `.iqmsg` files.

16.4.7 Logging Server Requests

To isolate some types of problems, especially problems with queries, log server requests.

You can enable request-level logging by either:

- Setting the `-zr` command line option when you start the server, or
- Calling the `sa_server_option` stored procedure, which overrides the current setting of the `-zr`.

Server requests are logged in `*.srvlog`. The `-zr` server startup option enables request-level logging of operations and sets the type of requests to log (SQL | HOSTVARS | PLAN | PROCEDURES | TRIGGERS | OTHER | BLOCKS | REPLACE | ALL | NONE). The `-zo` option redirects request-level logging information to a file separate from the regular log file. The `-zs` limits the size of this file.

> **i Note**
>
> If the size of the query text being written to the log exceeds the specified limit, the query text is not truncated and is logged in its entirety.

Once the database server is started, you can adjust the request log settings to log more or less information using `sa_server_option`. These commands enable request logging of a limited set of requests and redirect the output to the file `sqllog.txt`:

```
call sa_server_option('RequestLogging','SQL');
    call sa_server_option('RequestLogFile',
                          'sqllog.txt')
```

To disable request-level logging, use:

```
call sa_server_option('RequestLogging','NO');
```

To view the current settings for the SQL log file and logging level, execute:

```
select property('RequestLogFile'), property('RequestLogging');
```

To match `.iqmsg` log and the `-zr` server request log (`.srvlog`) entries using connection information, correlate connection information between the `.srvlog` and `.iqmsg` files.

> **i Note**
>
> SAP IQ version 15.1 modified the request log. Instead of fixed-format line prefixes, common information began being recorded as comma-delimited text. Where possible, times are recorded as "=" (meaning the same as the previous line) or +nnn (meaning nnn milliseconds after the previous line). Consequently, request logs are much smaller now than in versions earlier than SAP IQ 15.1.

In addition, more information is recorded in the request log. For queries, the information recorded is isolation level, number of rows fetched, and cursor type. For `INSERT`, `UPDATE`, and `DELETE` statements, the information recorded is number of rows affected and number of triggers fired.

Optionally, you can also choose to log statements executed within procedures and triggers.

You can select to record the short form of query plans in the request log. If procedure logging is enabled, plans for procedure statements are also recorded.

The following output shows an excerpt from the request log, when the server is started with the `-zr all` option. In this example, the user connects to the `iqdemo` database and executes `sp_iqstatus`.

There are several comma-separated fields in each line, and the first field indicates the time. Periodically, a full timestamp is output in the form:

```
MMdd hhmmss.sss
0523 095954.807,[,1000000001,sp_iq_mpx_init,16,iq utilities status 1
```

For lines after this line, for example, "`+13,C,1,UID=DBA`", the offset is from the previous line. In this case, "`+13`" means that about 13 milliseconds have passed since the last line. In some cases, "=" means approximately 0 milliseconds have elapsed since the last line.

The following shows an excerpt of the request log:

```
0523 095954.807,[,1000000001,sp_iq_mpx_init,16,iq
utilities status 1
+2,],1000000001,sp_iq_mpx_init,16
+1,[,1000000001,sp_iq_mpx_init,62,message STRING('IQ
Server ',@@servername,'.') to console
+2,],1000000001,sp_iq_mpx_init,62
taj% pg iqdemo.sqllog
0523 095954.807,[,1000000001,sp_iq_mpx_init,16,iq
utilities status 1
+2,],1000000001,sp_iq_mpx_init,16
+1,[,1000000001,sp_iq_mpx_init,62,message STRING('IQ
Server ',@@servername,'.') to console
+2,],1000000001,sp_iq_mpx_init,62
0523 100510.344,<,1,CONNECT
+13,C,1,UID=DBA
+83,>,1,CONNECT,1
+1,<,1,PREPARE,SELECT @@version, if 'A'<>'a' then 1
else 0 endif, isnull(property('IsIQ'),'NO'),
isnull(connection_property('odbc_distinguish_char_and_
varchar'),'Off'),
isnull(connection_property('odbc_describe_binary_as_va
rbinary'),'Off'), connection_property('charset'),
db_property('charset')
+1,>,1,PREPARE,65536
=,<,1,EXEC,65536
+79,P,1,[S]DUMMY<seq>
=,>,1,EXEC
+1,<,1,DROP_STMT,65536
=,>,1,DROP_STMT
=,<,1,PREPARE,SET TEMPORARY OPTION time_format =
```

```
'hh:nn:ss';SET TEMPORARY OPTION timestamp_format =
'yyyy-mm-dd hh:nn:ss.ssssss';SET TEMPORARY OPTION
date_format = 'yyyy-mm-dd';SET TEMPORARY OPTION
date_order = 'ymd';SET TEMPORARY OPTION isolation_level
= 0;
+1,>,1,PREPARE,65537
+1,<,1,EXEC,65537
=,[,1,*batch*,1,set temporary option time_format =
'hh:nn:ss'
+11,],1,*batch*,1
=,[,1,*batch*,1,set temporary option timestamp_format =
'yyyy-mm-dd hh:nn:ss.ssssss'
+11,],1,*batch*,1
+1,[,1,*batch*,1,set temporary option date_format =
'yyyy-mm-dd'
+11,],1,*batch*,1
=,[,1,*batch*,1,set temporary option date_order = 'ymd'+11,],1,*batch*,1
=,[,1,*batch*,1,set temporary option isolation_level = 0
+11,],1,*batch*,1
=,>,1,EXEC
```

In this section:

Request Log File Analysis [page 456]
Use the stored procedures `sa_get_request_profile` and `sa_get_request_times` to read the `-zr` log file and summarize the results.

16.4.7.1 Request Log File Analysis

Use the stored procedures `sa_get_request_profile` and `sa_get_request_times` to read the `-zr` log file and summarize the results.

`sa_get_request_profile` analyzes the request log to determine the execution times of similar statements, and summarizes the results in the global temporary table `satmp_request_profile`. For example:

```
call sa_get_request_profile('/sys1/users/jones/iqreqs1_zr.log');
select * from satmp_request_profile;
```

`sa_get_request_times` also analyzes the request log to determine statement execution times and summarizes the results in the global temporary table `satmp_request_time`. For example:

```
call sa_get_request_times('/sys1/users/jones/iqreqs1_zr.log');
select * from satmp_request_time;
```

For more information about request-level logging, see the start_iq `-zo` switch in *SAP IQ Utility Reference* > *start_iq Database Server Startup Utility*, and the `sa_server_option` system procedure in *SAP IQ SQL Reference*.

In this section:

Request logging [page 457]
Request logging logs individual requests received from, and responses sent to, an application.

16.4.7.1.1 Request logging

Request logging logs individual requests received from, and responses sent to, an application.

It is most useful for determining what the database server is being asked to do by the application.

Request logging is also a good starting point for performance analysis of a specific application when it is not obvious whether the database server or the client is at fault. You can use request logging to determine the specific request to the database server that might be responsible for problems.

> **i Note**
>
> All the functionality and data provided by the request logging feature is also available using diagnostic tracing. Diagnostic tracing also offers additional features and data.

Logged information includes such things as timestamps, connection IDs, and request type. For queries, it also includes the isolation level, number of rows fetched, and cursor type. For INSERT, UPDATE, and DELETE statements, it also includes the number of rows affected and number of triggers fired.

> **i Note**
>
> The request log can contain statements with obfuscated sensitive information. If you store passwords or keys in plain text to the server, then that information is logged. The only case when sensitive information is not obfuscated is a statement with a parsing error.

You can use the -zr server option to turn on request logging when you start the database server. You can redirect the output to a request log file for further analysis using the -zo server option. The -zn and -zs option let you specify the number of request log files that are saved and the maximum size of request log files. Additionally, you can use the sa_server_option system procedure to set up request logging on a server that is already running.

The sa_get_request_times system procedure reads a request log and populates a global temporary table (IQTMP16_request_time) with statements from the log and their execution times. For INSERT/UPDATE/DELETE statements, the time recorded is the time when the statements were executed. For queries, the time recorded is the total elapsed time from PREPARE to DROP (describe/open/fetch/close). Request times are not provided for cursors that are still open.

Analyze IQTMP16_request_time for statements that could be candidates for improvements. Statements that are inexpensive, but frequently executed, may represent performance problems.

You can use sa_get_request_profile to call sa_get_request_times and summarize IQTMP16_request_time into another global temporary table called IQTMP16_request_profile. This procedure also groups statements together and provides the number of calls, execution times, and so on.

> **⚠ Caution**
>
> If the log is being analyzed using the `tracetime.pl` Perl script, the max_client_statements_cached option should be set to 0 to disable client statement caching while the request log is captured.

Example

Output to the request log can be filtered to include only requests from a specific connection or from a specific database, using the sa_server_option system procedure. This can help reduce the size of the log when monitoring a database server with many active connections or multiple databases.

To filter according to a connection:

```
CALL sa_server_option( 'RequestFilterConn' , <connection-id> );
```

You can obtain `<connection-id>` by executing `CALL sa_conn_info( )`.

To filter according to a database:

```
CALL sa_server_option( 'RequestFilterDB' , <database-id> );
```

The `<database-id>` can be obtained by executing `SELECT CONNECTION_PROPERTY( 'DBNumber' )` when connected to that database. Filtering continues until explicitly reset, or until the database server is shut down.

To reset filtering, use either of the following two statements to reset filtering either by connection or by database:

```
CALL sa_server_option( 'RequestFilterConn' , -1 );
```

```
CALL sa_server_option( 'RequestFilterDB' , -1 );
```

To include host variable values in the request log:

- use the -zr server option with a value of hostvars
- execute the following:

  ```
  CALL sa_server_option( 'RequestLogging' , 'hostvars' );
  ```

The request log analysis procedure, sa_get_request_times, recognizes host variables in the log and adds them to the global temporary table IQTMP16_request_hostvar.

16.4.8 Connection for Collecting Diagnostic Information

The database option `DEDICATED_TASK` lets the DBA dedicate a request handling task to handle requests from a single connection.

This pre-established connection allows you to gather information about the state of the database server if it becomes otherwise unresponsive. See the `DEDICATED_TASK` option in *SAP IQ SQL Reference*.

16.4.9 Diagnosing Communications Issues

If your server returns a communication error on startup, you may want to set the `-z` command line option when you start the server.

This switch provides diagnostic information on communications links at server startup. Information is logged to standard output from where the server started and in the `srvlog` file.

16.5 Reporting Problems to Technical Support

If you cannot resolve a problem using the manuals or online help, the designated person should contact SAP Technical Support or the SAP subsidiary in your area.

Each SAP installation that has purchased a support contract has one or more designated people who are authorized to contact SAP Technical Support.

Technical Support needs information about your SAP IQ environment to resolve your problem.

In this section:

Collecting Diagnostic Information Using getiqinfo [page 459]
SAP IQ includes a script for collecting information that SAP Technical Support needs to diagnose problems.

Information Collected by getiqinfo [page 461]
The `getiqinfo` script collects information.

Correlating Connection Information Between the .srvlog and .iqmsg Files [page 462]
Technical Support may ask you to set the `-zr` option on the `start_iq` command in your configuration file.

16.5.1 Collecting Diagnostic Information Using getiqinfo

SAP IQ includes a script for collecting information that SAP Technical Support needs to diagnose problems.

The `getiqinfo` script collects information about the operating system environment, the SAP IQ environment, and log files.

Run this script before reporting a problem to SAP Technical Support. By doing so, you can help SAP staff resolve your issue more quickly, with less effort on your part.

`getiqinfo` is not designed for troubleshooting SAP IQ installations and does not provide on-site troubleshooting facilities. This script executes successfully only when the SAP IQ environment is properly set up and the server is running.

In this section:

Before You Run getiqinfo [page 460]

Collect information before running the `getiqinfo` script.

Running the getiqinfo Script [page 460]

On UNIX platforms, `getiqinfo` is a shell script. On Windows platforms, `getiqinfo.bat` is a batch script in the `IQ-16_1\bin64` directory. The `getiqinfo` script is installed with IQ Server; it is unavailable with the Network Client.

16.5.1.1 Before You Run getiqinfo

Collect information before running the `getiqinfo` script.

Before running the script, know the following information:

- Location of the database file.
- Full path of the configuration file used to start the server, if one is used.
- Full path of the `.iqmsg` file, if the SAP IQ message file has been renamed.

If possible, leave the SAP IQ server running, or start the server before running `getiqinfo`. This allows the script to collect internal database data that is available only when SAP IQ is running. The script does not automatically start the server.

The script runs with the same environment settings that are used to start the SAP IQ server. `getiqinfo` uses some IQ-specific environment variables to search for files.

The script places collected data in the current directory (where you start the program). Be sure you have enough space under that directory. The script does not prompt for an alternative, but you can modify the script to change the output location by resetting the `<DEST_DIR>` variable.

16.5.1.2 Running the getiqinfo Script

On UNIX platforms, `getiqinfo` is a shell script. On Windows platforms, `getiqinfo.bat` is a batch script in the `IQ-16_1\bin64` directory. The `getiqinfo` script is installed with IQ Server; it is unavailable with the Network Client.

Context

The steps vary for UNIX and Windows platforms.

Procedure

1. Start the script according to your platform:

Option	Description
UNIX	At the command prompt, in the `IQ-16_1/bin64` directory, type: `getiqinfo.sh`
Windows	Select **Start** > **Run** > `<install_path>\IQ-16_1\\bin64\getiqinfo.bat`.

2. As you are prompted, enter:
   - The directory of the database file. This is also the default location of the `.iqmsg` file, and the `stktrc*.iq` file on UNIX.
   - The base name of the database file (the file name without the `.db` suffix). This is also the default base name of the `.iqmsg` file.
   - Other directories to search for these files.
   - SAP IQ engine name (server name) and port number for this database server.
   - User ID and password of a user granted one of:
     - DROP CONNECTION system privilege
     - MONITOR system privilege
     - SERVER OPERATOR system privilege
   - The full path to the configuration file used to start the SAP IQ server, if one was used.
   - The full path to the output file in the `-zo` server option, if one was specified.

Results

The program also directs you to send the listed files to SAP IQ Technical Support.

16.5.2 Information Collected by getiqinfo

The `getiqinfo` script collects information.

- Type of hardware, amount of memory, CPU type, speed, number of CPUs
- Operating system (for example, Sun Solaris 2.10)
- Swap space size
- `IQ-16_1` version and EBF level, and Anywhere version
- Stack trace file for the date and time this problem occurred, named `stktrc-YYYYMMDD-HHMMSS_#.iq`, in the directory where you started the database server. (UNIX and Linux platforms only)
- Command or query that produced the error
- Message log file, named `dbname.iqmsg`, located, by default, in the directory where you started the database server
- Query plan (recorded in `.iqmsg` file; see the note below)
- Server logs
  - For UNIX, `IQ-16_1/logfiles/<servername>.000n.stderr` and `IQ-16_1/logfiles/<servername>.000n.srvlog`

    - On Windows platforms, if needed, you must restart the server and manually collect a copy of the console window
- Startup and connection option settings, from the configuration file (by default, `dbname.cfg`)
- Database option settings and output from `sa_conn_properties` (if the server is still running)

The following information is not collected by `getiqinfo`, but may be requested by Technical Support:

- Connectivity protocol used (for example, ODBC, JDBC, TDS)
- Open Client version
- Configuration type (single user or multi user)
- Front-end tool used (for example, Brio Query)
- Schema and indexes for the database
- Output from `sp_iqcheckdb` procedure

Query plan detail is collected automatically by `getiqinfo` if the options below are set. You can also collect this information manually, by setting the options and re-running the command that produced the error.

```
SET TEMPORARY OPTION QUERY_PLAN = 'ON'
```

```
SET TEMPORARY OPTION QUERY_DETAIL = 'ON'
```

The query plan is in the message log file. The default values for these options are QUERY_PLAN = ON and QUERY_DETAIL = OFF.

If you have performance problems, set the following option:

```
SET TEMPORARY OPTION QUERY_PLAN_AFTER_RUN = 'ON'
```

Setting this option enables Technical Support to see which steps in the query processing used the time.

16.5.3 Correlating Connection Information Between the .srvlog and .iqmsg Files

Technical Support may ask you to set the `-zr` option on the `start_iq` command in your configuration file.

Context

This server startup option sets the request logging level to track statements sent to the server. Parameters are ALL, NONE, or SQL. The option produces a log file named for the server with the suffix `.srvlog`.

In the SAP IQ message file `.iqmsg`, each connection to the server is identified by a connection handle. The `.iqmsg` message file records the errors, warnings, and tracing information for each connection.

To correlate the connection identifiers in the `.srvlog` and `.iqmsg` files to find relevant information.

Procedure

1. In the `.iqmsg` file, locate a connection of interest. For example:

   ```
   Connect:  SA connHandle: 1000000061
   ```

   These lines show the `.iqmsg` log file contents for this connection:

   ```
   16:14:59. 0000000062 Connect:  SA connHandle: 1000000061
             SA connID: 31  IQ connID: 0000000062  User: DBA
   03/17 16:15:00. 0000000062 Cmt 12064
   03/17 16:15:00. 0000000062 PostCmt 0
   03/17 16:15:00. 0000000000 Disconnect:  SA connHandle: 1000000061
                   SA connID: 31  IQ connID: 0000000062  User: DBA
   ```

2. Isolate all of the lines for the connection by searching the `.srvlog` file for the number that follows "SA connHandle" in the `.iqmsg` file.

   For example, search the `.srvlog` file for "1000000061":

   ```
   16:14:59. [,1000000061,sp_iqdbspace,48,select
   str_replace(dbspaceName,'"',null) into dbspaceName_literal
   03/17 16:14:59. P,1000000061,[S][0]DUMMY<seq>
   03/17 16:14:59. ],1000000061,sp_iqdbspace,48
   03/17 16:14:59. P,1000000061,[1]ISYSIQDBFILE<seq> JNL dbf<ISYSDBFILE>
   JNL ISYSDBSPACE<ISYSDBSPACE>
   03/17 16:14:59. [,1000000061,sp_iqdbspace,58,execute immediate with
   quotes on   'iq utilities main into iq_dbspace_temp dbspace info
   ' || dbspaceName
   03/17 16:14:59. P,1000000061,[S]INSERT ROWS
   03/17 16:14:59. P,1000000061,[S]INSERT ROWS
   03/17 16:14:59. P,1000000061,[S]INSERT ROWS
   03/17 16:14:59. P,1000000061,[S]INSERT ROWS03/17 16:14:59. P,1000000061,
   [S]INSERT ROWS
   03/17 16:14:59. ],1000000061,sp_iqdbspace,58
   03/17 16:14:59. [,1000000061,sp_iqdbspace,60,select d.dbspace_name as
   DBSpaceName,   min(SegType) as DBSpaceType,...
   03/17 16:15:00. ],1000000061,sp_iqdbspace,60
   03/17 16:15:00. P,1000000061,Work[ Sort[ GrByH[ dbf<seq> JNL
   ISYSIQDBSPACE<ISYSIQDBSPACE> JNL ISYSDBSPACE<ISYSDBSPACE> JH*
   iq_dbspace_temp<seq> ] ] ] : ISYSIQPARTITIONCOLUMN<seq> :
   idx<seq> : tab<seq>
   03/17 16:15:00. [,1000000061,sp_iqdbspace,105,drop table
   dbo.iq_dbspace_temp
   03/17 16:15:00. ],1000000061,sp_iqdbspace,105
   03/17 16:15:00. P,1000000061,[1]Work[ Sort[ sp_iqdbspace<call> ] ]
   ```

   The connection handle in this example is 1000000061.

16.6 Checklist: Information for Technical Support

You can run the `getiqinfo` script to collect information.

Information Requested	Value
SAP IQ version (for example 16.1 GA or SP number)	

Information Requested	Value
`sp_iqlmconfig` output	
Type of hardware	
Amount of memory	
Number of CPUs	
Operating system name and version (for example, Microsoft Windows 2008 Service Pack 1)	
Operating system patch level	
Front-end tool used (for example, Business Objects Crystal Reports)	
Connectivity protocol used (for example, ODBC, JDBC, TDS)	
Open Client version	
Configuration type (single node or multiplex)	
Message log file (`dbname.iqmsg`)	
Server log files (`server.nnnn.srvlog` and `server.nnnn.stderr`)	
Stack trace file (`stktrc-YYYYMMDD-HHNNSS_#.iq`)	
Command or query that produced the error	
Start up option settings	
Connect option settings	
Database option settings	
Schema and indexes for the database	
`sp_iqstatus` output	
Query plan: `set` options (Query_Plan, Query_Detail, Query_Plan_After_Run, Query_Plan_As_Html, Query_Plan_As_Html_Directory, Query_Timing), re-run command or query	
Screen snapshot of the problem, if possible.	

17 Connection and Communication Parameters Reference

Connection parameters establish and describe connections from client applications to a database.

In this section:

Connection Parameters [page 465]
Connection parameters are included in connection strings.

Network Communications Parameters [page 512]
If you experience problems with client/server network communications, you can set a number of command line parameters for both the client and the server. These parameters enable you to work around peculiarities of different network protocol implementations.

17.1 Connection Parameters

Connection parameters are included in connection strings.

You can enter connection parameters in:

- An application's connection string
- An ODBC data source
- The SAP IQ Connect dialog

After each parameter name, the short form appears in square brackets. You can use the short form as an abbreviation in connect commands.

The ODBC configuration dialog and the SAP IQ Connect dialog for Windows operating systems share a common format. Some parameters correspond to check boxes and fields in these dialogs, while others can be entered in the text box on the **Advanced** tab.

Usage Notes

Connection parameters are case-insensitive.

The usage for each connection parameter describes the circumstances under which the parameter. Common usage entries include:

- Embedded databases – when SAP IQ is used as an embedded database, the connection starts a server and loads the database. When the application disconnects from the database, the database is unloaded and the server stops.
- Network servers – when SAP IQ is used as a network server, the client application must locate a server already running on the network and connect to a database.

You can use the `dbping` utility to test connection strings. For examples, see the *SAP IQ Utility Reference* > *dbping Database Administration Utility*.

Boolean (true or false) arguments are

- If true – YES, ON, 1, or TRUE
- If false – NO, OFF, 0, or FALSE

You can obtain the connection parameters used by the interface library from these places, in order of precedence:

- Connection string
- SQLCONNECT environment variable
- Data sources

The server name must be composed of characters in the range 1 to 127 of the ASCII character set. There is no such limitation on other parameters.

These rules govern the priority of parameters:

- The entries in a connection string are read left to right. If the same parameter is specified more than once, the last one in the string applies.
- If a string contains a DSN or FILEDSN entry, the profile is read from the configuration file, and the entries from the file are used if they are not already set. For example, if a connection string contains a data source name and sets some of the parameters contained in the data source explicitly, then in case of conflict, the explicit parameters are used.

In this section:

AppInfo Connection Parameter [page 468]
Helps administrators identify the origin of particular client connections from a database server.

AutoPreCommit Connection Parameter [page 470]
Forces each statement to commit before execution.

AutoStart (ASTART) Connection Parameter [page 471]
Prevents a database server from being started if no connection is found.

AutoStop (ASTOP) Connection Parameter [page 472]
Prevents unloading of a database as soon as there are no more open connections to it.

CharSet (CS) Connection Parameter [page 473]
Specifies the character set to be used on this connection.

CompressionThreshold (COMPTH) connection parameter [page 474]
Increases or decreases the size limit at which packets are compressed.

CommBufferSize (CBSize) Connection Parameter [page 475]
Sets the maximum size of communication packets, in bytes. Use k to specify units of kilobytes.

CommBufferSpace (CBSpace) Connection Parameter [page 476]
Specifies the amount of space to allocate on startup for network buffers, in kilobytes.

CommLinks (LINKS) Connection Parameter [page 477]
Specifies client-side network communications links.

ConnectionName (CON) Connection Parameter [page 479]
Names a connection to make switching to it easier in multi-connection applications.

ConnectionPool (CPOOL) Connection Parameter [page 480]
Controls the behavior of client connection pooling.

DatabaseFile (DB) Connection Parameter [page 481]
Specifies the target database file for a load or connection, when starting a database.

DatabaseName (DBN) Connection Parameter [page 482]
Specifies a loaded database to which a connection needs to be made. Use when connecting to a database that is already running.

DatabaseSwitches (DBS) Connection Parameter [page 484]
Provides database-specific switches when starting a database.

DataSourceName (DSN) Connection Parameter [page 485]
Tells the ODBC driver manager or Embedded SQL library where to look in the `.odbc.ini` file (on UNIX), or `odbc.ini` file or registry (on Windows), to find ODBC data source information.

DBKEY Connection Parameter [page 486]
Starts an encrypted database with a connect request.

DisableMultiRowFetch (DMRF) Connection Parameter [page 486]
Turns off multi-record fetches across the network.

EngineName (ENG) Connection Parameter [page 487]
Specifies a running database server to which you want to connect. `ServerName` is a synonym for `EngineName`.

EncryptedPassword (ENP) Connection Parameter [page 488]
Provides a password, stored in an encrypted fashion in a data source.

Encryption (ENC) connection parameter [page 489]
Specifies the encoding for packets sent between the client application and the database server. This may be no encoding, simple obfuscation, or TLS (Transport Layer Security) encryption.

Escape (ESCAPE) Connection Parameter [page 492]
Allows you to specify the escape character used in the LIKE clause of SQL statements generated by the ODBC driver when returning a list of tables or column.

FileDataSourceName (FILEDSN) Connection Parameter [page 493]
Tells the client library that there is an ODBC file data source holding information about the database to which you want to connect.

Idle (IDLE) Connection Parameter [page 494]
Specifies the idle timeout period of the connection.

Integrated (INT) Connection Parameter [page 495]
Uses the integrated login facility.

Language (LANG) Connection Parameter [page 496]
Specifies the language of the connection.

LazyClose (LCLOSE) Connection Parameter [page 497]
Causes the `CLOSE <cursor-name>` database request to queue and be sent to the server with the next database request. This eliminates a network request each time a cursor is closed.

LivenessTimeout (LTO) Connection Parameter [page 498]
Controls the termination of connections when they are no longer intact.

LogFile (LOG) Connection Parameter [page 499]
Sends client error messages and debugging messages to a file.

LogicalServer (LS) Connection Parameter [page 500]
Specifies the target logical server for a connection.

LoginRedirection (REDIRECT) Connection Parameter [page 501]
Controls login redirection at the connection level.

NewPassword (NEWPWD) connection parameter [page 502]
Allows users to change passwords, even if they have expired, without DBA intervention.

NodeType Connection Parameter [page 503]
Specifies the server role for a logical server member node.

Password (PWD) Connection Parameter [page 504]
Provides a password for the connection.

PrefetchBuffer (PBUF) Connection Parameter [page 505]
Allows the user to set the maximum amount of memory, in bytes, for buffering rows.

PrefetchRows (PROWS) Connection Parameter [page 507]
The `PrefetchRows` connection parameter sets the maximum number of rows to prefetch when querying the database.

RetryConnectionTimeout (RetryConnTO) connection parameter [page 508]
Instructs the client library (DBLIB, ODBC, ADO, and so on) to keep retrying the connection attempt, as long as the server is not found, for the specified period of time.

ServerName Connection Parameter [page 509]
Synonym for `EngineName` (ENG) connection parameter.

StartLine (START) Connection Parameter [page 509]
Starts a database server running from an application.

Unconditional (UNC) Connection Parameter [page 510]
Stops a server using `dbstop` even when there are connections to the server.

Userid (UID) Connection Parameter [page 511]
Specifies the user ID with which you log on to the database.

17.1.1 AppInfo Connection Parameter

Helps administrators identify the origin of particular client connections from a database server.

Usage

Anywhere

Default

Empty string

Remarks

This connection parameter is sent to the database server from Embedded SQL, ODBC, or OLE DB clients, as well as `DBISQLC` on UNIX. It is not available from Open Client.

The parameter consists of a generated string that holds information about the client process, such as the IP address of the client machine, the operating system it is running on, and so on. The string is associated in the database server with the connection, and you can retrieve it using:

```
SELECT connection_property( 'AppInfo' )
```

Clients can also specify their own string, which is appended to the generated string. The `AppInfo` property string is a sequence of semicolon-delimited `key=<value>` pairs. The valid keys are as follows:

- API – DBLIB, ODBC, OLEDB, or ADO.NET (ODBC is returned of the iAnywhere JDBC driver)
- APPINFO – string entered, if you specified AppInfo in the connection string
- EXE – name of the client executable (Windows only)
- HOST – host name of the client machine
- IP – IP address of the client machine
- OS – operating system name and version number (for example, Sun Solaris 2.9)
- PID – process ID of the client
- THREAD – thread ID of the client
- VERSION – version of the connection protocol in use, including major and minor values, and a build number (for example 9.0.1.1549)
- TIMEZONEADJUSTMENT – number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the connection

If you specify a debug log file in your client connection parameters, the APPINFO string is added to the file.

Examples

- Connect to the demo database from Interactive SQL (the iAnywhere JDBC driver is used by default):

  ```
  dbisql nogui -c "uid=DBA;pwd=<database>" -host MachineName -port 1234
  ```

  View the application information:

  ```
  SELECT connection_property('AppInfo')
  ```

  The result is as follows (in a single string):

  ```
  IP=12.345.67.899;
  HOST=<MachineName>;
  OSUSER=sap;
  OS='SunOS 5.10 Generic_144489_04';
  EXE='/SAP/IQ16/shared/JRE-7_0_1_64BIT/bin/java';
  PID=0x52af;
  THREAD=0x14;
  VERSION=16.0.0.290;
  API=iAnywhereJDBC;
  TIMEZONEADJUSTMENT=-240
  ```

- Connect to the default database from Interactive SQL, appending your own information to the AppInfo property:

  ```
  isql -U DBA -P <password> -S MachineName:1234 --appname MyISQL
  ```

  View the application information:

  ```
  SELECT connection_property('AppInfo')
  ```

  The result is as follows (in a single string):

  ```
  HOST=MachineName;
  PID=21155;
  EXE=MyISQL
  ```

17.1.2 AutoPreCommit Connection Parameter

Forces each statement to commit before execution.

Syntax

```
AutoPreCommit={ YES | NO }
```

Usage

ODBC

Default

NO

Remarks

By default, statements issue a commit after execution. When `AutoPreCommit = YES`, commit statements are issued before each select statement, so that users can always see the latest version of all database objects.

Example

You can set the `AutoPreCommit` option to YES to turn on commit before execution or No to turn it off. Set this option in the `.odbc.ini` file or on the Advanced tab of the Connect dialog.

For example, this causes each statement to commit before execution:

```
[Sample DSN]
DatabaseFile=c:\Program Files\SAP\IQ-16_1\demo\iqdemo.db
AutoPreCommit=Y
UserID=DBA
Password=<database>
```

17.1.3 AutoStart (ASTART) Connection Parameter

Prevents a database server from being started if no connection is found.

Syntax

```
{ AutoStart | ASTART } = { YES | NO }
```

Usage

Anywhere

Default

YES

Remarks

By default, if no server is found during a connection attempt, and a database file is specified, then a database server is started on the same machine. You can turn this behavior off by setting the `AutoStart` parameter off in the connection string.

Example

The following data source fragment prevents a database server from being started if no network server is located:

```
[My Demo Database]
DatabaseFile=c:\SAP\IQ-16_1\demo\iqdemo.db
Autostart=No
UserID=DBA
ENG=network_server
```

17.1.4 AutoStop (ASTOP) Connection Parameter

Prevents unloading of a database as soon as there are no more open connections to it.

Syntax

```
{ AutoStop | ASTOP } = { YES | NO }
```

Usage

Embedded databases

Default

YES

Remarks

By default, any server that is started from a connection string is stopped when there are no more connections to it. Also, any database that is loaded from a connection string is unloaded as soon as there are no more connections to it. This behavior is equivalent to `Autostop=YES`.

If you supply `Autostop=NO`, any database that you start in that connection is not unloaded when there are no more connections to it. As a consequence, the database server is not shut down either.

The `AutoStop` parameter is used only if you are connecting to a database that is not currently running. It is ignored if the database is already loaded.

Example

The following Windows connection profile prevents the database from being unloaded when the connection is dropped:

```
[Sample Embedded Database]
DatabaseFile=c:\SAP\IQ-16_1\demo\iqdemo.db
Autostop=No
UserID=DBA
```

17.1.5 CharSet (CS) Connection Parameter

Specifies the character set to be used on this connection.

Syntax

```
{ CharSet | CS }=<STRING>
```

Usage

Anywhere

Default

The locale character set

Remarks

If you supply a value for `CharSet`, the specified character set is used for the current connection. `CharSet=none` disables character set conversion for the connection.

When unloading data, you can specify the character set using the `CharSet` connection parameter.

To avoid lossy character-set conversions, avoid setting the CharSet connection parameter when using Unicode client APIs. Unicode client APIs include ADO.NET, OLE DB, and the iAnywhere JDBC driver. ODBC is also a Unicode client API when the wide (Unicode) functions are used.

17.1.6 CompressionThreshold (COMPTH) connection parameter

Increases or decreases the size limit at which packets are compressed.

Syntax

```
{ CompressionThreshold | COMPTH }=<size>[ k ]
```

Usage

Anywhere except TDS. Only applies to compressed connections.

Allowed values

size

This integer specifies the size limit at which packets are compressed. The default value is in bytes, but you can use k to specify units of kilobytes. If both the client and database server specify different compression threshold settings, the client setting applies. The minimum supported value is 1 byte, and the maximum supported value is 32767 bytes. Values less than 80 bytes are not recommended.

Default

120

If no CompressionThreshold value is set, the compression threshold value is controlled by the setting on the server, which defaults to 120 bytes.

Remarks

Changing the compression threshold can help performance of a compressed connection by allowing you to only compress packets when compression will increase the speed at which the packets are transferred.

When compression is enabled, individual packets may or may not be compressed, depending on their size. For example, the database server does not compress packets smaller than the compression threshold, even if communication compression is enabled. As well, small packets (less than about 100 bytes) usually do not compress at all. Since CPU time is required to compress packets, attempting to compress small packets could actually decrease performance.

Lowering the compression threshold value may improve performance on very slow networks, while raising the compression threshold may improve performance by reducing CPU. However, since lowering the compression threshold value will increase CPU usage on both the client and server, a performance analysis should be done to determine whether changing the compression threshold is beneficial.

Example

Connect, with a compression threshold of 100 bytes.

```
CompressionThreshold=100
```

17.1.7 CommBufferSize (CBSize) Connection Parameter

Sets the maximum size of communication packets, in bytes. Use k to specify units of kilobytes.

Syntax

```
{ CommBufferSize | CBSIZE }=size[ k ]
```

Usage

Anywhere

Allowed Values

`<size>`

Specifies the maximum size of communication packets. The default value is in bytes, but you can use k to specify units of kilobytes. The minimum value of `CommBufferSize` is 500 bytes, and the maximum is 65535 bytes.

Default

If no CommBufferSize value is set, the `CommBufferSize` is controlled by the setting on the server, which defaults to 7300 bytes.

Remarks

The protocol stack sets the maximum size of a packet on a network. If you set the `CommBufferSize` to be larger than that permitted by your network, the communication packets are broken up by the network software. The default size is a multiple of the standard ethernet TCP/IP maximum packet size (1460 bytes).

A larger packet size may improve performance for multi-row fetches and fetches of larger rows, but it also increases memory usage for both the client and the server.

If `CommBufferSize` is not specified on the client, the connection uses the server's buffer size. If `CommBufferSize` is specified on the client, the connection uses the `CommBufferSize` value.

Using the `-p` database server option to set the `CommBufferSize` causes all clients that do not specify their own `CommBufferSize` to use the size specified by the `-p` database server option.

Example

This example sets the buffer size to 1460 bytes:

```
...
CommBufferSize=1460
...
```

Alternatively, you can set this parameter by entering its value in the `Buffer size` text box of the `Network` tab of the connection window.

17.1.8 CommBufferSpace (CBSpace) Connection Parameter

Specifies the amount of space to allocate on startup for network buffers, in kilobytes.

Syntax

```
{ CommBufferSpace | CBSpace }=<INTEGER>
```

Usage

Anywhere

Allowed Values

Integer

Default

10

Remarks

`CommBufferSpace` is a global setting, for all connections.

Examples

The following profile fragment instructs the network library to allocate 200 KB for network buffers on startup:

```
...
CBSpace=200
...
```

You can set this parameter by entering its value in the **Buffer space** text box of the **Network** tab of the connection window.

17.1.9 CommLinks (LINKS) Connection Parameter

Specifies client-side network communications links.

Syntax

```
{ CommLinks | LINKS }=<STRING>
```

Usage

Anywhere

Allowed Values

String

Default

Use only the shared memory communications link to connect.

Remarks

If you specify `CommLinks=ALL`, the client searches for a server using all available communication protocols. Since there may be an impact on performance if you specify `CommLinks=ALL`, use this setting only when you do not know which protocol to use.

If you specify one or more protocols in the `CommLinks` connection parameter, the client uses the named communication protocols, in the order specified, to search for a network database server. A connection error appears and the connection attempt aborts if the connection fails to connect using a specified protocol, even if there are protocols remaining in the list to try.

If you do not specify a `CommLinks` connection parameter, the client searches for a server on the current machine only, and only using a shared memory connection. This is the default behavior, and is equivalent to `CommLinks=ShMem`. The shared memory protocol is used for communication between a client and server running under the same operating system on the same machine.

Available values of the `CommLinks` parameter are:

- SharedMemory (ShMem) – start the shared memory protocol for same-machine communication. This is the default setting. The client tries shared memory first if it appears in a list of protocols, regardless of the order in which protocols appear.
- ALL –attempt to connect using the shared memory protocol first, followed by all remaining and available communications protocols. Use this setting if you are unsure of which communication protocol(s) to use.
- TCPIP – start the TCP/IP communications link. TCP/IP is supported on all operating systems.

Each of these values can have additional network communications parameters supplied.

Some of the reasons you may want to use a specific protocol, instead of `ALL`:

- The network library starts slightly faster if you do not start unnecessary network links.
- Connecting to the database may be faster.
- You must specify the link explicitly to tune the broadcast behavior of a particular protocol by providing additional network communications parameters.

Additional network communications parameters may be provided for each link, to tune the broadcast behavior of the link.

The `CommLinks` parameter corresponds to the database server -x `iqsrv16` server option. By default, the network server starts all available protocols, which is equivalent to `-x ALL`.

Examples

- The following connection string fragment starts the TCP/IP protocol only:

```
CommLinks=tcpip
```

- The following connection string fragment starts the shared memory protocol and searches for the database server over shared memory:

  ```
  CommLinks=tcpip(HOST=kangaroo),shmem
  ```

  If the search fails, it then starts the TCP/IP port searching for the host `kangaroo` in addition to servers on the immediate TCP/IP network.

17.1.10 ConnectionName (CON) Connection Parameter

Names a connection to make switching to it easier in multi-connection applications.

Syntax

```
{ ConnectionName | CON }=<STRING>
```

Usage

Not available for ODBC

Default

No connection name

Remarks

An optional parameter, providing a name for the particular connection you are making. You may leave this unspecified unless you are going to establish more than one connection, and switch between them.

The Connection Name is not the same as the data source name.

Examples

Connect, naming the connection FirstCon:

```
CON=FirstCon
```

17.1.11 ConnectionPool (CPOOL) Connection Parameter

Controls the behavior of client connection pooling.

Syntax

```
{ ConnectionPool | CPOOL }= { NO | YES
   [ ( [ Timeout=<timeout-sec>; ] [ MaxCached=<max-cached-conn> ] ) ] }
```

Usage

All platforms except non-threaded UNIX clients.

Allowed Values

- `<timeout-sec>` – the idle timeout period, in seconds, of the connection pool. The default value is 60 seconds. Cached connections that are not reused within the time specified by the timeout-sec value are disconnected and are no longer available for reuse.
- `<max-cached-conn>` – the maximum number of cached connections from each application. The default value is five connections. A connection is cached if it is disconnected and the maximum number of connections specified by the max-cached-conn value has not been reached. The connection is reinitialized, and the cached connection remains connected to the database server even though the application has disconnected it.

Default

YES

Remarks

Connection pooling may improve the performance of applications that make multiple, brief connections to the database server. When a connection is disconnected it is automatically cached and may be reused when the application reconnects. For a connection to be pooled, the connection name can be different, but all other connection parameters must be identical.

17.1.12 DatabaseFile (DB) Connection Parameter

Specifies the target database file for a load or connection, when starting a database.

Syntax

```
{ DatabaseFile | DB }=<STRING>
```

Usage

Embedded databases

Allowed Values

String

Default

None

Remarks

To connect to an already running database, use the `DatabaseName` parameter.

SAP IQ requires the `DatabaseName` parameter and the database file name to connect under special circumstances.

`DatabaseName` loads and connects to a specific database file that is not already running on a database server.

- If a database is loaded with a name that is the same as the `DatabaseName` parameter, but without the `.db` extension, the connection is made to that database instead.
- If the file name does not include an extension, a file of name `.db` is looked for.
- The path of the file is relative to the working directory of the database server. If you start the server from the command prompt, the working directory is the directory you are in when you enter the command. If you start the server from an icon or shortcut, it is the working directory specified in the icon or shortcut. It is recommended that you supply a complete path and file name.
- If you specify both the database file and the database name, the database file is ignored and is not used to try to connect to a database that is already running.

You can also use UNC file names.

> ⚠ Caution
>
> The database file must be on the same machine as the database server. Managing a database file that is located on a network drive might lead to file corruption.

See also *DatabaseName (DBN) Connection Parameter*.

Example

To load and connect to the demo database (installed in directory `C:\Program Files\SAP\IQ-16_1\demo` on Windows), use the following `DatabaseName` parameter:

```
DBF=C:\Program Files\SAP\IQ-16_1\demo\iqdemo.db
```

Related Information

DatabaseName (DBN) Connection Parameter [page 482]

17.1.13 DatabaseName (DBN) Connection Parameter

Specifies a loaded database to which a connection needs to be made. Use when connecting to a database that is already running.

> Syntax
>
> ```
> { DatabaseName | DBN }=<STRING>
> ```

Usage

Running network servers

Allowed Values

String

Default

None

Remarks

To connect to a database that is not already running, use the `DatabaseFile` (DBF) parameter.

When a database is started on a server, it is assigned a database name, either by the administrator using the `-n` option, or by the server using the base of the file name with the extension and path removed.

If the database you want to connect to is already running, specify the database name rather than the database file.

A connection occurs only if the name of the running database matches the name that is specified in the `DatabaseName` parameter.

> **i** Note
>
> If you specify both the database name and database file, the database file is ignored and is not used to try to connect to a database that is already running.

Examples

To start a database file named `cities.db` and rename the database Kitchener, you can use:

```
start_iq cities.db -n Kitchener
```

Assuming you have run the above command, you can successfully connect to the running database named Kitchener:

```
DBN=Kitchener
```

Alternatively:

```
DBN=Kitchener;DBF=cities.db
```

However, this fails to connect to the database named Kitchener:

```
DBF=cities.db
```

See also *DatabaseFile (DB) Connection Parameter*.

Related Information

DatabaseFile (DB) Connection Parameter [page 481]

17.1.14 DatabaseSwitches (DBS) Connection Parameter

Provides database-specific switches when starting a database.

Syntax

```
{ DatabaseSwitches | DBS }=<STRING>
```

Usage

Connecting to a server when the database is not loaded.

Default

No switches

Remarks

Supply `DatabaseSwitches` only if you are connecting to a database that is not currently running. When the server starts the database specified by `DatabaseFile`, the server uses the supplied `DatabaseSwitches` as command line options to determine startup options for the database.

Only database switches can be supplied using this parameter. Server switches must be supplied using the START connection parameter.

Examples

The following UNIX command, entered all on one line, connects to the default database server, loads the database file IQ-16_1`/demo/iqdemo.db` (DBF parameter), names it as `my_db` (DBS parameter) and connects to the database of that name (DBN parameter):

```
dbcollat -c "uid=DBA;pwd=SQL;dbf=/IQ-16_1/demo/iqdemo.db;
dbn=my_db;dbs=-n my_db" /tmp/temp.col
```

17.1.15 DataSourceName (DSN) Connection Parameter

Tells the ODBC driver manager or Embedded SQL library where to look in the `.odbc.ini` file (on UNIX), or `odbc.ini` file or registry (on Windows), to find ODBC data source information.

Syntax

```
{ DataSourceName | DSN }=<STRING>
```

Usage

Anywhere

Default

None

Remarks

It is common practice for ODBC applications to send only a data source name to ODBC. The ODBC driver manager and ODBC driver locate the data source, which contains the remainder of the connection parameters. In SAP IQ, Embedded SQL applications can also use ODBC data sources to store connection parameters.

DSN does not support verbose form of keyword.

Example

This parameter uses a data source name:

```
DSN=Dynamo Demo
```

17.1.16 DBKEY Connection Parameter

Starts an encrypted database with a connect request.

Syntax

```
DBKEY=<STRING>
```

Usage

On database startup. Not used when connecting to a running database.

Default

By default, databases are not encrypted.

Remarks

You must specify this parameter when you start an encrypted database with a connect request.

Examples

The following profile fragment uses the encryption key “is!seCret” to connect to a strongly encrypted database named `marvin.db` that is already running:

```
...
UID=dba;PWD=MyIQD@tabase;DBF=marvin.db;DBKEY='is!seCret'
...
```

17.1.17 DisableMultiRowFetch (DMRF) Connection Parameter

Turns off multi-record fetches across the network.

Syntax

```
{ DisableMultiRowFetch | DMRF }= { YES | NO }
```

Usage

Anywhere

Default

NO

Remarks

By default, when the database server gets a simple fetch request, the application asks for extra rows. You can disable this behavior by enabling this parameter.

Setting the `DisableMultiRowFetch` parameter to YES is equivalent to setting the PREFETCH option to OFF.

Example

This connection string fragment prevents prefetching:

```
DMRF=Yes
```

17.1.18 EngineName (ENG) Connection Parameter

Specifies a running database server to which you want to connect. `ServerName` is a synonym for `EngineName`.

Syntax

```
{ EngineName | ENG }=<STRING>
```

Usage

Network servers

Allowed Values

String

Default

The default local database server

Remarks

To connect to a network server, supply an `EngineName`. In the Connect dialog, and in the ODBC Administrator, this is the `Server Name` field.

The server name is interpreted according to the character set of the client machine. Multi-byte characters are not recommended in server names.

Names must be a valid identifier. Long server names are truncated to different lengths depending on the protocol.

For database server naming restrictions, see the *SAP IQ Utility Reference*.

Protocol	Truncation Length
UNIX shared memory	31 bytes
Non-UNIX shared memory	40 bytes
TCP/IP	40 bytes

Examples

This example connects to a server named Guelph:

```
ENG=Guelph
```

17.1.19 EncryptedPassword (ENP) Connection Parameter

Provides a password, stored in an encrypted fashion in a data source.

Syntax

```
{ EncryptedPassword | ENP }=<STRING>
```

Usage

Anywhere (DSN and FILEDSN connection parameters do not support verbose form of keyword)

Allowed Values

String

Default

None

Remarks

Data sources are stored on disk as a file or in the registry. Storing passwords on disk may present a security problem. For this reason, when you enter a password into a data source, it is stored in an encrypted form.

If you specify both `Password` and `EncryptedPassword`, `Password` takes precedence.

See also *Password (PWD) Connection Parameter*.

Related Information

Password (PWD) Connection Parameter [page 504]

17.1.20 Encryption (ENC) connection parameter

Specifies the encoding for packets sent between the client application and the database server. This may be no encoding, simple obfuscation, or TLS (Transport Layer Security) encryption.

Syntax

```
{ Encryption | ENC }=
  { NONE
  | SIMPLE
  | TLS [ (
      [ FIPS={ ON | OFF }; ]
      [ TRUSTED_CERTIFICATE={ <public-certificate-filename> | * } ; ]
      [ CERTIFICATE_COMPANY=<organization>; ]
      [ CERTIFICATE_NAME=<common-name>; ]
```

```
    [ CERTIFICATE_UNIT=<organization-unit>; ]
    [ SKIP_CERTIFICATE_NAME_CHECK={ ON | OFF } ]
    [ IDENTITY=<identity-file> ]
    [ IDENTITY_PASSWORD=<password> ]
    [  ALLOW_EXPIRED_CERTS= { ON | OFF } ]
) ]
 }
```

Usage

TLS: supported for TCP/IP only

NONE or SIMPLE: anywhere

Allowed values

NONE

Accepts communication packets that are not encrypted.

SIMPLE

Accepts communication packets that are encoded with simple obfuscation. This value is supported on all platforms and on previous versions of SAP IQ. Simple obfuscation does not provide database server verification, strong encryption, or other features of Transport Layer Security.

If the database server accepts simple obfuscation, but does not accept unencrypted communication, then any non-TDS connection attempts using no encryption automatically use simple obfuscation.

Starting the database server with `-ec SIMPLE` tells the database server to accept only connections using simple obfuscation. TLS connections requesting RSA encryption or FIPS-certified RSA encryption fail, and connections requesting no encryption use simple obfuscation.

TLS

Accepts communication packets that are encrypted using TLS. For FIPS-certified TLS, specify FIPS=ON. FIPS-certified TLS uses a separate certified library, but is compatible with SAP IQ 9.0.2 or later database servers using non-certified TLS.

When initiating TLS connections, the client libraries will check the host name of the database server against the certificate provided by that server. This check will only happen if none of the certificate_name, certificate_company, or certificate_unit options are specified, or the skip_certificate_name_check option is not enabled. If any of certificate_name, certificate_company, or certificate_unit are specified, only those options are verified. The skip_certificate_name_check option disables the host name check when enabled.

The host names or IP addresses are derived from the subjectAltName (Subject Alternative Name or SAN) extension and the Common Name (CN) field. The SAN may contain multiple host names with wild cards. For example, a Google certificate might include *.google.com, *.google.ca, and *.android.com. Thus, www.google.ca is a valid host name.

The client supports the following arguments to verify the field values in the database server's certificate, whether you are using a public certificate or a certificate from the operating system's certificate store.

- CERTIFICATE_COMPANY
- CERTIFICATE_NAME
- CERTIFICATE_UNIT

TLS and TLS(TRUSTED_CERTIFICATE=*) are equivalent, and specifying either causes the software to use the operating system certificate store and trust every certificate listed in the store.

Specify SKIP_CERTIFICATE_NAME_CHECK=ON to disable the host name check. Enabling this option may prevent the client from fully authenticating the database server.

Use the IDENTITY_PASSWORD option to specify a password for the encryption certificate.

Specify ALLOW_EXPIRED_CERTS=ON for the client libraries to trust a root certificate and accept a database server certificate that is either expired or is not yet valid. The default is OFF.

Default

NONE

Remarks

This parameter can be used to encode communications between client applications and the database server using Transport Layer Security (TLS) or simple obfuscation.

FIPS-certified encryption requires a separate license.

Use the CONNECTION_PROPERTY system function to retrieve the encryption settings for the current connection:

```
SELECT CONNECTION_PROPERTY ( 'Encryption' );
```

Example

The following connection string fragment connects to a database server using Transport Layer Security (RSA encryption) and a specified certificate file that is accepted even if it is expired:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(FIPS=OFF;TRUSTED_CERTIFICATE=rsaroot.
crt;ALLOW_EXPIRED_CERTS=ON)"
```

The following connection string fragment connects to a database server using simple obfuscation:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=simple"
```

The following two connection string fragments are equivalent and connect to a database server using Transport Layer Security (RSA encryption) and a certificate from the operating system certificate store:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(FIPS=OFF;TRUSTED_CERTIFICATE=*)"
```

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS"
```

The following connection string fragment verifies that `myrootcert.crt` is at the root of the database server's certificate's signing chain and that the common name field of the certificate is set to MyCertificateName:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(TRUSTED_CERTIFICATE=myrootcert.crt;CE
RTIFICATE_NAME=MyCertificateName)"
```

The following connection string fragment verifies that `myrootcert.crt` is at the root of the database server's certificate's signing chain. Since no other TLS certificate option is specified, the host name in the database server's certificate must match the host name being connected to:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(TRUSTED_CERTIFICATE=myrootcert.crt)"
```

The following connection string fragment verifies that `myrootcert.crt` is at the root of the database server's certificate's signing chain, but does no other verification:

```
"HOST=myhost;SERVER=myserver;ENCRYPTION=TLS(TRUSTED_CERTIFICATE=myrootcert.crt;SK
IP_CERTIFICATE_NAME_CHECK=ON)"
```

17.1.21 Escape (ESCAPE) Connection Parameter

Allows you to specify the escape character used in the LIKE clause of SQL statements generated by the ODBC driver when returning a list of tables or column.

Syntax

```
{ Escape | ESCAPE }=<STRING>
```

Remarks

The `Escape` parameter is supported only by SAP IQ.

By default, the ODBC driver uses the tilde (~) as an escape character, but some applications assume that the escape character is a backslash (\). For example, the default escape character for characters stored as hexadecimal codes and symbols is a backslash (\), so `\x0A` represents the line feed character.

Examples

To use the exclamation mark as the escape character, enter:

```
ESCAPE='!'
```

17.1.22 FileDataSourceName (FILEDSN) Connection Parameter

Tells the client library that there is an ODBC file data source holding information about the database to which you want to connect.

Syntax

```
{ FileDataSourceName | FILEDSN }=<STRING>
```

Usage

Anywhere

Allowed Values

String

Default

None

Remarks

File data sources hold the same information as ODBC data sources stored in a registry. File data sources can be easily distributed to end users, so that connection information does not have to be reconstructed on each machine.

Both ODBC and Embedded SQL applications can use file data sources.

Examples

This is a data source description held in a file data source:

```
[Sample File Data Source]
ENG=iqdemo
DBA=DBA
```

```
PWD=SQL
```

17.1.23 Idle (IDLE) Connection Parameter

Specifies the idle timeout period of the connection.

Syntax

```
{ Idle | IDLE }=<INTEGER>
```

Usage

Anywhere except with Tabular Data Stream (TDS) and Shared Memory connections. Shared Memory and TDS connections (including jConnect) ignore the SAP IQ `Idle` (`IDLE`) connection parameter.

Allowed Values

Integer

Default

Value of `-ti`

Remarks

The `Idle` (`IDLE`) connection parameter applies only to the current connection. You can have multiple connections on the same server that are set to different timeout values.

If no connection idle timeout value is set, the idle timeout value is controlled by the setting on the server, which defaults to 4400 minutes when set by `start_iq`. In case of a conflict between timeout values, the connection timeout value supersedes any server timeout value whether specified or unspecified.

Optionally, the relevant server command line parameters can be included for both Idle and Liveness Timeout (`-ti` and `-tl`, respectively).

See Also

`-ti iqsrv16` server option in *SAP IQ Utility Reference* > *start_iq Database Server Startup Utility*.

Example

This connection string fragment sets the timeout value for this connection to 10 minutes:

```
"ENG=myeng;LINKS=tcpip;IDLE=10"
```

17.1.24 Integrated (INT) Connection Parameter

Uses the integrated login facility.

Syntax

```
{ Integrated | INT }={ YES | NO }
```

Usage

Anywhere

Allowed Values

YES, NO

Default

NO

Remarks

The `Integrated` parameter has the following settings:

- An integrated login is attempted. If the connection attempt fails and the LOGIN_MODE option is set to Mixed (deprecated), a standard login is attempted.
- This is the default setting. No integrated login is attempted.

For a client application to use an integrated login, the server must be running with the LOGIN_MODE database option set to Mixed (deprecated) or Integrated.

Examples

This data source fragment uses an integrated login:

```
INT=yes
```

17.1.25 Language (LANG) Connection Parameter

Specifies the language of the connection.

Syntax

```
{ Language | LANG }=<STRING>
```

Usage

Anywhere

Allowed Values

The two-letter combination representing a language. For example, setting `LANG=DE` sets the default language to German.

Default

The language specified by (in order) the IQLANG environment variable or the installer.

Remarks

This connection parameter establishes the language for the connection. Any errors or warnings from the server are delivered in the specified language, assuming that the server supports the language.

If no language is specified, the default language is used.

This connection parameter affects only the connection. Messages returned from SAP SQL Anywhere tools and utilities appear in the default language, while messages returned from the server appear in the connection's language.

17.1.26 LazyClose (LCLOSE) Connection Parameter

Causes the `CLOSE <cursor-name>` database request to queue and be sent to the server with the next database request. This eliminates a network request each time a cursor is closed.

Syntax

```
{ LazyClose | LCLOSE }={ YES | NO }
```

Usage

Anywhere

Allowed Values

YES, NO

Default

NO

Remarks

When this parameter is enabled, cursors are not actually closed until the next database request. Any isolation level 1 cursor stability locks still apply to the cursor while the `CLOSE <cursor-name>` database request is queued.

Enabling this option can improve performance in the following conditions:

- If your network exhibits poor latency
- If your application sends many cursor open and close requests

In rare circumstances, canceling the next request after the `CLOSE <cursor-name>` database request can leave the cursor in a state where it appears to be closed on the client side, but is not actually closed on the server side. Subsequent attempts to open another cursor with the same name fail. Do not use `LazyClose` if your application cancels requests frequently.

17.1.27 LivenessTimeout (LTO) Connection Parameter

Controls the termination of connections when they are no longer intact.

Syntax

```
{ LivenessTimeout | LTO }=<INTEGER>
```

Usage

Network server on TCP/IP communications protocols.

All platforms except non-threaded UNIX applications.

Allowed Values

Integer, in seconds

Default

120

If no `LivenessTimeout` value is set, the liveness timeout is controlled by the setting on the server, which defaults to 120 seconds.

Remarks

A liveness packet is periodically sent across a client/server TCP/IP communications protocol to confirm that a connection is intact. If the client runs for the liveness timeout period without detecting a liveness request or response packet, communication is severed.

Liveness packets are sent when a connection has not sent any packets for between one third and two thirds of the `LivenessTimeout` value.

When communication is severed, the client machine forgets the address of the server. It looks the address up next time there is a connection to the server from that machine, dropping all current connections to that server.

When there are more than 200 connections to a server, the server automatically calculates a higher `LivenessTimeout` value based on the stated `LivenessTimeout` value. This enables the server to more efficiently handle a large number of connections.

Alternatively, you can set this parameter by entering its value in the LivenessTimeout text box of the Network tab of the ODBC Configuration dialog.

Example

This sets a `LivenessTimeout` value of 60 seconds:

```
LTO=60
```

17.1.28 LogFile (LOG) Connection Parameter

Sends client error messages and debugging messages to a file.

Syntax

```
{ LogFile | LOG }=<STRING>
```

Usage

Anywhere

Allowed Values

String

Remarks

To save client error messages and debugging messages in a file, use the `LogFile` parameter.

If the file name includes a path, it is relative to the current working directory of the client application.

The `LogFile` connection parameter is connection-specific, so from a single application you can set different `LogFile` arguments for different connections.

Examples

This command line fragment specifies that client messages for this connection should be sent to the file `error.log` in the current working directory for the client:

```
...
LogFile=error.log
...
```

17.1.29 LogicalServer (LS) Connection Parameter

Specifies the target logical server for a connection.

Syntax

```
{ LogicalServer | LS }=<STRING>
```

Usage

Anywhere

Allowed Values

String

Remarks

Use this parameter to explicitly specify the target logical server, enabling SAP IQ to redirect a connection from one multiplex node to another.

If `LogicalServer` is unspecified in the connection string, the setting of default logical server in the user's login policy serves as the logical server context.

17.1.30 LoginRedirection (REDIRECT) Connection Parameter

Controls login redirection at the connection level.

Syntax

```
{ LoginRedirection | REDIRECT }={ ON | OFF }
```

Usage

Login redirection at the logical server level must be enabled for this parameter to take effect.

Allowed Values

ON, OFF

Default

ON

Remarks

Certain applications may need to target specific nodes within a logical server.

When you specify `REDIRECT=OFF`, no login redirection takes place for that connection. The connection fails if the node of the initial connection cannot satisfy the connection requirements of target logical server and requested server role.

For example, you could disable login redirection at the connection level to target nodes within the logical server that contain data in SAP SQL Anywhere tables, which is not shared between nodes.

17.1.31 NewPassword (NEWPWD) connection parameter

Allows users to change passwords, even if they have expired, without DBA intervention.

Syntax

```
{ NewPassword | NEWPWD }={ <password-string> | * }
```

Usage

The client library prompting for a new password is only supported on Microsoft Windows.

Allowed values

password-string

If the user provides a new password, the database server authenticates the user ID and password and attempts to change the password before the login_procedure option is called. This process allows the user to change an expired password without the involvement of a DBA. If you have set the verify_password_function option, the new password is verified.

*

On Microsoft Windows, if you use the special value *, the client library prompts for a new password during a connection attempt only if the existing password has expired. The user must provide their existing password, provide their new password, and confirm their new password. When the user completes the fields and clicks **OK**, the old password is authenticated and the database server attempts to change the password. If you have set the verify_password_function option, the new password is verified. The process of verifying if a user's password has expired, prompting for a password, and authenticating and changing the password occurs with a single connect call to the client library.

Default

The password is not changed, and the client library does not prompt for a new password.

Remarks

If you are authenticating with an Integrated or Kerberos login, the original password is not validated, the database server ignores the new password value and the password is not changed, and access to the database is denied.

If you are authenticating using LDAP user authentication (LDAPUA) or PAM user authentication (PAMUA), this connection parameter has no effect.

This connection parameter is very effective when you implement a login policy using the password_life_time or password_expiry_on_next_login options. Alternatively, you can implement a password expiry policy by having the login_procedure signal the `Password has expired` error.

A user receives a `Password has expired error` if their environment does not support password prompting. In a Microsoft Windows environment, the prompt window might not correctly prevent interaction with the calling application's window (it may not be modal or have the correct parent window) if the calling application has multiple top-level windows or if the application's top level windows are minimized.

In a Windows environment, if you use the ODBC SQLDriverConnect function and the DriverCompletion argument is anything other than SQL_DRIVER_NOPROMPT, the connection prompts for a new password if the password has expired. The connection might prompt for a new password in OLE DB when the DBPROP_INIT_PROMPT property is anything other than DBPROMPT_NOPROMPT. Both cases function as if the NewPassword=* connection parameter was specified.

Example

The following connection string changes the password of user Test1 when they connect:

```
"UID=Test1;PWD=welcome;NEWPWD=hello"
```

In a Windows environment, the following connection string prompts the user Test1 for a new password when the existing password expires:

```
"UID=Test1;PWD=welcome;NEWPWD=*"
```

17.1.32 NodeType Connection Parameter

Specifies the server role for a logical server member node.

Syntax

```
NodeType={ READER | WRITER | ANY }
```

Usage

Anywhere

Allowed Values

- READER – applications that execute user-defined functions may need to connect to a member node with the READER role.
- WRITER – applications that execute statements (such as `INSERT`, `UPDATE`, and `DELETE`) may need to connect to a member node with the WRITER role.
- ANY – most read-only applications (that do not execute any UDF) can connect to any member node of the logical server.

Default

ANY

Remarks

A logical server can have both reader and writer nodes as members. This parameter allows an application to connect to a member node with a specific role. You can specify the desired role as READER, WRITER, or ANY.

SAP IQ ignores the `NodeType` connection parameter when you log in with the SERVER logical server context.

17.1.33 Password (PWD) Connection Parameter

Provides a password for the connection.

Syntax

```
{ Password | PWD }=<STRING>
```

Usage

Anywhere

Default

None

Remarks

Every database user has a password. The password must be supplied for the user to connect to the database. By default, the password has the same case-sensitivity as the data, and by default SAP IQ databases are case-sensitive.

The password parameter is not encrypted. If you are storing passwords in a connection profile, use the `EncryptedPassword` parameter. The SAP IQ ODBC configuration tool uses encrypted parameters.

If you specify both `Password` and `EncryptedPassword` are specified, `Password` takes precedence.

See also *EncryptedPassword (ENP) Connection Parameter*.

Examples

The following connection string fragment supplies the user ID `DBA` and password `SQL`:

```
uid=DBA;pwd=SQL
```

Alternatively, you can set these parameters in the **User ID** and **Password** text boxes in the connection window.

Related Information

EncryptedPassword (ENP) Connection Parameter [page 488]

17.1.34 PrefetchBuffer (PBUF) Connection Parameter

Allows the user to set the maximum amount of memory, in bytes, for buffering rows.

Syntax

```
{ PrefetchBuffer | PBUF }=<INTEGER>[ k | m ]
```

Usage

Anywhere

Allowed Values

Integer, in bytes, between 64 KB and 8 MB

Default

512 (KB)

Remarks

The `PrefetchBuffer` connection parameter controls the memory allocated on the client to store prefetched rows. The value is in bytes, but you can optionally use `k` or `m` to specify units in kilobytes or megabytes. This connection parameter accepts values between 64 KB and 8 MB. In some circumstances, increasing the number of rows prefetched from the database server by the client can improve query performance. You can increase the number of rows prefetched using the `PrefetchRows` and `PrefetchBuffer` connection parameters.

Increasing the `PrefetchBuffer` connection parameter increases the amount of memory used to buffer GET DATA requests. This may improve performance for some applications that process many GET DATA (SQLGetData) requests.

See also *PrefetchRows (PROWS) Connection Parameter*.

Example

- The following example determines if the PrefetchBuffer memory limit is reducing the number of prefetched rows:

  ```
  ...prefetchrows=100;logfile=c:\ client.txt
  ```

- The following example increases the memory limit to 256 KB:

  ```
  ...prefetchrows=100;prefetchbuffer=256
  ```

Related Information

PrefetchRows (PROWS) Connection Parameter [page 507]

17.1.35 PrefetchRows (PROWS) Connection Parameter

The `PrefetchRows` connection parameter sets the maximum number of rows to prefetch when querying the database.

Syntax

```
{ PrefetchRows | PROWS }=<INTEGER>
```

Usage

Anywhere

Default

10 (200 when using the .NET Data Provider)

Remarks

Increasing the number of rows prefetched from the database server by the client can improve performance on cursors that do only fetch relative 0 or 1, with either single row or wide fetches. Wide fetches include embedded SQL array fetches and ODBC block fetches.

Improvements occur particularly under the following conditions:

- The application fetches many rows (several hundred or more) with very few absolute fetches.
- The application fetches rows at a high rate, and the client and server are on the same machine or connected by a fast network.
- Client/server communication is over a slow network, such as a dial-up link or wide area network.

The number of rows prefetched is limited both by the `PrefetchRows` connection parameter and the `PrefetchBuffer` parameter, which limits the memory available for storing prefetched rows.

The maximum number of rows that can be prefetched is 1000.

See also *PrefetchBuffer (PBUF) Connection Parameter*.

Example

The following connection string fragment sets the number of prefetched rows to 100:

```
...PrefetchRows=100;...
```

Related Information

PrefetchBuffer (PBUF) Connection Parameter [page 505]

17.1.36 RetryConnectionTimeout (RetryConnTO) connection parameter

Instructs the client library (DBLIB, ODBC, ADO, and so on) to keep retrying the connection attempt, as long as the server is not found, for the specified period of time.

Syntax

```
{ RetryConnectionTimeout | RetryConnTO }=<timeout-value>
```

Usage

Anywhere

Allowed values

timeout-value

The value specified by this connection is a timeout, in seconds. It is not a counter of the number of times to retry the connection attempt. The default value of zero indicates that the connection attempt should only be tried once.

Default

0

Remarks

There is a half-second delay between iterations, and the retries only occur if the connection attempt failed because the database server was not found. Any other error is returned immediately. If the database server is not found, the connection attempt will take at least as long as the time specified by the RetryConnectionTimeout connection parameter.

The default TCP timeout is 5 seconds, so if your connection string contains a value for RetryConnTO that is less than 5, for example `Host=<host-name>;RetryConnTO=3`, then the connection attempt still takes 5 seconds.

Example

The following connection string fragment tells the client library to continue to retry the connection attempt for at least 5 seconds:

```
...RetryConnTO=5;...
```

17.1.37 ServerName Connection Parameter

Synonym for `EngineName` (ENG) connection parameter.

See *EngineName (ENG) Connection Parameter*.

Related Information

EngineName (ENG) Connection Parameter [page 487]

17.1.38 StartLine (START) Connection Parameter

Starts a database server running from an application.

Syntax

```
{ StartLine | START }=<STRING>
```

Usage

Embedded databases

Default

None

Remarks

Supply a `StartLine` parameter only if you are connecting to a database server that is not currently running. The `StartLine` parameter is a command line to start a server.

Example

The following data source fragment starts a database server with a cache of 32 MB:

```
StartLine=dbeng6 -c 32M iqdemo.db
```

17.1.39 Unconditional (UNC) Connection Parameter

Stops a server using `dbstop` even when there are connections to the server.

Syntax

```
{ Unconditional | UNC }={ YES | NO }
```

Usage

Anywhere

Default

No

Remarks

The `dbstop` command line utility shuts down a database server. If you specify `Unconditional=YES` in the connection string, the server is shut down even if there are active connections. If `Unconditional` is not set to `YES`, the server shuts down only if there are no active connections.

Example

The following command line shuts down the server unconditionally:

```
dbstop -c "uid=DBA;pwd=SQL;eng=server-name;unc=yes"
```

17.1.40 Userid (UID) Connection Parameter

Specifies the user ID with which you log on to the database.

Syntax

```
{ Userid | UID }=<STRING>
```

Usage

Anywhere (DSN and FILEDSN connection parameters do not support verbose form of keyword)

Default

None

Remarks

You must always supply a user ID when connecting to a database. The user ID is not case-sensitive, and is unaffected by the setting of the Case Respect database property.

Examples

The following connection string fragment supplies the user ID `DBA` and password `SQL`:

```
uid=DBA;pwd=SQL
```

17.2 Network Communications Parameters

If you experience problems with client/server network communications, you can set a number of command line parameters for both the client and the server. These parameters enable you to work around peculiarities of different network protocol implementations.

Supply the network communication parameters on the server or client command line as in the following example:

```
start_iq -x tcpip(PARM1=value1;PARM2=value2;. . .),...
```

From the client side, communication parameters are entered as in this example:

```
CommLinks=tcpip(PARM1=value1;PARM2=value2;. . .),...
```

If they include spaces, the network communication parameters must be enclosed in quotation marks to be parsed properly:

```
start_iq -x "tcpip(PARM1=value 1;PARM2=value 2;...),..."
start_iq -x "tcpip(PARM1=value1;PARM2=value2;...)"
```

In UNIX, quotation marks are required if more than one parameter is given, because UNIX interprets the semicolon as a command separator.

Boolean parameters are turned on with any of YES, ON, TRUE, or 1, and are turned off with any of NO, OFF, FALSE, or 0. The parameters are case-insensitive.

Enter the commands to set the communication parameters on a single line; you can also include them in a configuration file, then use the @ server or client command line switch to invoke the configuration file.

TCP/IP, HTTP, and HTTPS Communications Parameters

The parameters currently available for TCP/IP, HTTP, and HTTPS are as follows.

TCP/IP	HTTP and HTTPS
Broadcast [BCAST]	Identity
BroadcastListener [BLISTENER]	Identity_Password
ClientPort [CPORT]	DatabaseName [DBN]
DLL	LocalOnly [LOCAL]
DoBroadcast [DOBROAD]	LogFile [LOG]
Host [IP]	LogMaxSize [LSize]
LocalOnly [LOCAL]	LogOptions [LOpt]

TCP/IP	HTTP and HTTPS
LDAP [LDAP]	LogFormat [LF]
MyIP [ME]	MaxConnections [MaxConn]
ReceiveBufferSize [RCVBUFSZ]	MaxRequestSize [MaxSize]
SendBufferSize [SNDBUFSZ]	MyIP [ME]
ServerPort [PORT]	ServerPort [PORT]
TDS	Timeout [TO]
Timeout [TO]	
VerifyServerName [VERIFY]	

In this section:

Broadcast (BCAST) Communication Parameter [page 514]
Specifies a broadcast address.

BroadcastListener (BLISTENER) Communication Parameter [page 515]
Controls broadcast listening for this port.

ClientPort (CPort) Communication Parameter [page 516]
Designates the port number on which the client application communicates using TCP/IP. You may specify a single port number, or a combination of individual port numbers and ranges of port numbers.

DatabaseName (DBN) Communication Parameter [page 517]
Specifies the name of a database to use when processing web requests, or uses the REQUIRED or AUTO keyword to specify whether database names are required as part of the URI.

DoBroadcast (DBROAD) Communication Parameter [page 518]
Performs a broadcast to search for servers.

Host (IP) Communication Parameter [page 520]
Specifies additional machines outside the immediate network to be searched by the client library.

Identity Communication Parameter [page 521]
Specifies the name of an identity file.

Identity_Password Communication Parameter [page 523]
Specifies the password for the encryption certificate.

LDAP Communication Parameter [page 524]
Allows clients running over a WAN or through a firewall to find servers without specifying the IP address.

LocalOnly (LOCAL) Communication Parameter [page 525]
Allows a client to choose to connect only to a server on the local machine, if one exists.

LogFile (LOG) Communication Parameter [page 526]
Specifies the name of the file to which the database server is to write information about web requests.

LogFormat (LF) Communication Parameter [page 526]
Controls the format of messages written to the log file and which fields appear in them.

LogMaxSize (LSIZE) protocol option [page 527]
Controls the maximum size of the message log file where the database server writes information about web requests.

LogOptions (LOPT) Communication Parameter [page 528]
Controls which categories of messages are logged.

MaxConnections (MAXCONN) Communication Parameter [page 530]
Specifies the number of simultaneous connections accepted by the server.

MaxRequestSize (MAXSIZE) protocol option [page 530]
Specifies the size of the largest request the database server can accept.

MyIP (ME) Communication Parameter [page 531]
Controls whether the client attempts to determine addressing information.

PreFetchOnOpen Communication Parameter [page 533]
Sends a prefetch request with a cursor open request, thereby eliminating a network request to fetch rows each time a cursor is opened.

ReceiveBufferSize (RCVBUFSZ) Communication Parameter [page 534]
Sets the size for a buffer used by the TCP/IP protocol stack. You may want to increase the value if LOB performance over the network is important.

SendBufferSize (SNDBUFSZ) Communication Parameter [page 534]
Sets the size for a buffer used by the TCP/IP protocol stack.

ServerPort (PORT) Communication Parameter [page 535]
In the case of the database server, the `ServerPort` option designates the port number on which to communicate using TCP/IP. In a data source, the `ServerPort` option informs the client of the port or ports on which database servers are listening for TCP/IP communication.

Sessions Communication Parameter [page 537]
Sets the maximum number of clients that can communicate with the server at one time through a single LAN adapter.

TDS Communication Parameter [page 537]
Controls whether the server allows TDS connections.

Timeout (TO) Communication Parameter [page 538]
Specifies the length of time, in seconds, to wait for a response when establishing communications and when disconnecting.

VerifyServerName (VERIFY) Communication Parameter [page 540]
Specifies whether the server name is verified when connecting to this host.

17.2.1 Broadcast (BCAST) Communication Parameter

Specifies a broadcast address.

Syntax

```
{ Broadcast | BCAST }=<STRING>
```

Usage

TCP/IP

Allowed Values

String, in the form of an IP address)

Default

Broadcasts to all addresses on the same subnet.

Remarks

The default broadcast address is created using the local IP address and subnet mask. The subnet mask indicates which portion of the IP address identifies the network, and which part identifies the host.

For a subnet of 10.24.98.x, with a mask of 255.255.255.0, the default broadcast address is 10.24.98.255.

When specifying an IPv6 address on a Windows platform, use the interface identifier. UNIX platforms support both interface identifiers and interface names in IPv6 addresses. The interface identifier is required on Linux (kernel 2.6.13 and later).

Example

The following example tells the client to broadcast only on interface number 2 when using IPv6:

```
LINKS=tcpip(BROADCAST=ff02::1%2)
```

17.2.2 BroadcastListener (BLISTENER) Communication Parameter

Controls broadcast listening for this port.

Syntax

```
{ BroadcastListener | BLISTENER }={ YES | NO }
```

Usage

TCP/IP, server side

Allowed Values

YES, NO

Default

YES

Remarks

NO turns listening off for this port. Using `-sb 0` is the same as specifying `BroadcastListener=NO` on TCP/IP.

Example

The following example starts a server that accepts TCP/IP connections that use BroadcastListener=NO:

```
start_iq -x tcpip(BroadcastListener=NO)
```

17.2.3 ClientPort (CPort) Communication Parameter

Designates the port number on which the client application communicates using TCP/IP. You may specify a single port number, or a combination of individual port numbers and ranges of port numbers.

Syntax

```
{ ClientPort | CPort }=<INTEGER>
```

Usage

TCP/IP (client side only)

Default

Assigned dynamically per connection by the networking implementation. If you do not have firewall restrictions, do not use this parameter.

Remarks

This option is provided for connections across firewalls, as firewall software filters according to TCP/UDP port. Do not use this parameter unless you need to for firewall reasons.

Specify a list or a range of port numbers to make multiple connections using a given data source or connect string. If you specify a single port number, your application can maintain only one connection at a time. In fact, even after closing the one connection, there is a short timeout period during which no new connection can be made using the specified port. When you specify a list or range of port numbers, the application keeps trying port numbers until it finds one to which it can successfully bind.

Examples

- The following string makes a connection from an application using port 6000 to a server named my_server using port 5000:

  ```
  CommLinks=tcpip{ClientPort=6000;ServerPort=5000}; ServerName=my_server
  ```

- The following string makes a connection from an application that can use ports 5050 through 5060, as well as ports 5040 and 5070, for communicating with a server named my_server using the default server port:

  ```
  CommLinks=tcpip{ClientPort=5040,5050-5060,5070};ServerName=my_server
  ```

17.2.4 DatabaseName (DBN) Communication Parameter

Specifies the name of a database to use when processing web requests, or uses the REQUIRED or AUTO keyword to specify whether database names are required as part of the URI.

Syntax

```
{ DatabaseName | DBN }={ AUTO | REQUIRED | <database-name> }
```

Usage

HTTP, HTTPS

Allowed Values

AUTO, REQUIRED, <database-name>

Default

AUTO

Remarks

Setting this parameter results in the following:

- REQUIRED – the URI must specify a database name.
- AUTO – the URI may specify a database name, but does not need to do so. If the URI contains no database name, the default database on the server processes web requests. Since the server must guess whether or not the URI contains a database name when set to AUTO, you should design your web site so as to avoid ambiguity.
- The name of a database – that database is used to process all web requests. The URI must not contain a database name.

Example

The following command starts two databases, but permits only one of them to be accessed via HTTP:

```
start_iq -xs http(dbn=web) iqdemo.db web.db
```

17.2.5 DoBroadcast (DBROAD) Communication Parameter

Performs a broadcast to search for servers.

Syntax

```
{ DoBroadcast | DBROAD }={ { ALL | NONE | DIRECT } | { YES | NO } }
```

Usage

TCP/IP (all platforms)

Allowed Values

- Client side – ALL, NONE, DIRECT
- Server side – YES, NO

Default

ALL

Remarks

Client usage:

- With `DoBroadcast=ALL` (formerly `DoBroadcast=YES`) a broadcast is performed to search for a server. The broadcast goes first to the local subnet. If `HOST=` is specified, broadcast packets are also sent to each of the hosts. For TCP, all broadcast packets are UDP packets.
- With `DoBroadcast=DIRECT` (formerly `DoBroadcast=NO`), no broadcast is performed to the local subnet to search for a database server. Broadcast packets are sent only to the hosts listed in the HOST (IP) communication parameter. If you specify `DoBroadcast=DIRECT`, the HOST (IP) communication parameter is required.
- Specifying `DoBroadcast=NONE` causes no UDP broadcasts to be used. A TCP/IP connection is made directly with the HOST/PORT specified, and the server name is verified. With TCP/IP, you can choose not to verify the server name by setting the VerifyServerName (VERIFY) communication parameter to `NO`. The HOST (IP) communication parameter is a required parameter, while the ServerPort (PORT) communication parameter is optional.
- DIRECT and NONE, you must specify the server host with the HOST option.

Server usage:

- Setting `DoBroadcast=NO` prevents the database server from broadcasting to find other servers with the same name. This is useful in certain rare circumstances, but is not generally recommended.

See also *Host (IP)*, *LocalOnly (LOCAL)*, *ServerPort (PORT)*, and *VerifyServerName (VERIFY)* communication parameters.

Examples

- The following command starts a client without broadcasting to search for a database server. Instead, the server is looked for only on the computer named `silver`:

  ```
  dbisql -x tcpip(DOBROADCAST=DIRECT;HOST=silver) iqdemo
  ```

- On UNIX, the options must be enclosed in quotation marks:

  ```
  dbisql -x "tcpip(DOBROADCAST=DIRECT;HOST=silver)" iqdemo
  ```

Related Information

Host (IP) Communication Parameter [page 520]
LocalOnly (LOCAL) Communication Parameter [page 525]
ServerPort (PORT) Communication Parameter [page 535]
VerifyServerName (VERIFY) Communication Parameter [page 540]

17.2.6 Host (IP) Communication Parameter

Specifies additional machines outside the immediate network to be searched by the client library.

Syntax

```
{ Host | IP }=<STRING>
```

Usage

TCP/IP (all platforms) server and client sides

Allowed Values

String

Default

No additional machines

Remarks

On the server, the search is carried out to avoid starting a server with a duplicate name.

For TCP/IP, the `<hostname>` or a dot-separated IP address may be used.

The server prints this addressing information during startup if the `-z` switch is used. In addition, the application writes this information to its logfile if `LogFile` is specified (`Debug` is set to TRUE).

You can use a semicolon-separated list of addresses to search for more than one machine. Also, you can append a port number to an IP address using a colon as separator. Alternatively, you can specify the host and server ports explicitly, as in `<Host=nnn.nn.nnn.nnn;ServerPort=pppp>`.

`IP` and `HOST` are synonyms when using TCP/IP.

When specifying an IPv6 address on a Windows platform, use the interface identifier. UNIX platforms support both interface identifiers and interface names in IPv6 addresses. The interface identifier is required on Linux (kernel 2.6.13 and later).

See also *DoBroadcast (DBROAD) Communication Parameter*.

Examples

- The following connection string fragment instructs the client to look on the machines “kangaroo” and 197.75.209.222 (port 2369) to find a database server called `iqdemo`:

  ```
  ...ENG=iqdemo CommLinks=tcpip(IP=kangaroo;IP=197.75.209.222:2369)
  ```

- For UNIX, quotation marks are required around the TCP/IP options:

  ```
  dbisql -x "tcpip(HOST=kangaroo;HOST=197.75.209.222)" iqdemo
  ```

- The following connection string fragment instructs the client to look on the machines my_server and kangaroo to find a database server. A connection is attempted to the first host that responds:

  ```
  dbisql -c
  "UID=DBA;PWD=MyIQD@tabase;LINKS=tcpip(HOST=my_server,kangaroo;PORT=2639)"
  ```

Related Information

DoBroadcast (DBROAD) Communication Parameter [page 518]

17.2.7 Identity Communication Parameter

Specifies the name of an identity file.

Syntax

```
Identity=<identity-file>
```

Usage

HTTPS

Allowed Values

STRING

Default

None

Remarks

When you use transport-layer security, the identity file contains the public certificate and its private key, and for certificates that are not self-signed, the identity file also contains all the signing certificates, which includes, among other things, the encryption certificate. The password for this certificate must be specified with the`Identity_Password` parameter.

See also *Identity_Password Communication Parameter*.

Example

This example starts a database server that requires web connections to use a particular encryption certificate:

```
iqsrv16 -xs https(Identity=cert.file;Identity_Password=secret) ...
```

Related Information

Identity_Password Communication Parameter [page 523]

17.2.8 Identity_Password Communication Parameter

Specifies the password for the encryption certificate.

Syntax

```
Identity_Password=<STRING>
```

Usage

HTTPS

Allowed Values

STRING

This string specifies the password for an encryption certificate

Default

None

Remarks

When you use transport-layer security, this option specifies the password that matches the password for the encryption certificate specified by the `Identity` communication parameter.

See also *Identity Communication Parameter*.

Example

Start a database server that requires web connections to use a particular encryption certificate.

```
iqsrv16 -xs https(Identity=cert.file;Identity_Password=secret) ...
```

Related Information

Identity Communication Parameter [page 521]

17.2.9 LDAP Communication Parameter

Allows clients running over a WAN or through a firewall to find servers without specifying the IP address.

Syntax

```
LDAP={ YES | NO | <filename>}
```

Usage

Allows clients running over a WAN or through a firewall to find servers without specifying the IP address. It also allows the Locate utility (`dblocate`) to find such servers.

TCP/IP (server-side only)

Allowed Values

YES, NO, or `<filename>`

Default

ON

The default value for `<filename>` is `asaldap.ini`

Description

Having the database server register itself with an LDAP server allows clients (and the Locate utility [`dblocate`]) to query the LDAP server.

Specifying `LDAP=<filename>` turns LDAP support on and uses the specified file as the configuration file. Specifying `LDAP=YES` turns LDAP support on and uses `saldap.ini` as the configuration file.

LDAP is only used with TCP/IP, and only on network servers.

17.2.10 LocalOnly (LOCAL) Communication Parameter

Allows a client to choose to connect only to a server on the local machine, if one exists.

Syntax

```
{ LocalOnly | LOCAL }={ YES | NO }
```

Usage

TCP/IP, HTTP, HTTPS

Allowed Values

YES, NO

Default

NO

Remarks

If no server with the matching server name is found on the local machine, no server is autostarted.

The `LocalOnly` communication parameter is useful only if you set `DoBroadcast` to ALL (the default).

`LocalOnly=YES` uses the regular broadcast mechanism, except that broadcast responses from servers on other machines are ignored.

You can use the `LocalOnly` communication parameter with the server to restrict connections to the local machine. Connection attempts from remote machines do not find this server, and the Locate [`dblocate`] utility do not see this server. Running a server with the `LocalOnly` communication parameter set to YES allows the network server to run as a personal server without experiencing connection or CPU limits.

17.2.11 LogFile (LOG) Communication Parameter

Specifies the name of the file to which the database server is to write information about web requests.

Syntax

```
{ LogFile | LOG }=<filename>
```

Usage

HTTP, HTTPS

Allowed Values

STRING

Default

None

17.2.12 LogFormat (LF) Communication Parameter

Controls the format of messages written to the log file and which fields appear in them.

Syntax

```
{ LogFormat | LG }=<STRING>
```

Usage

HTTP, HTTPS

Allowed Values

Format string

Default

@T - @W - @I - @P - "@M @U @V" - @R - @L - @E

Description

If the message string contains any of the following codes, the current values are substituted for the codes as each message is written:

- The @ character
- Date and time that processing of the request started, unless the request cannot be queued due to an error
- Date and time that the client connected
- Name of the database associated with the request
- Text of the error message, if an error occurred
- Date and time that processing of the request finished
- IP address of the client
- Length of the response, in bytes, including headers and body
- HTTP request method
- Listener port associated with the request
- Date and time that the request was queued for processing, unless the request could not be queued due to an error
- Status code and description of the HTTP response
- HTTP status code
- Date and time that the current log entry was written
- Requested URI
- Requested HTTP version
- Time taken to process the request (@F – @B), or 0.000 if the request was not processed due to an error

17.2.13 LogMaxSize (LSIZE) protocol option

Controls the maximum size of the message log file where the database server writes information about web requests.

Syntax

```
{ LogMaxSize | LSIZE }=<size>[ k | m | g ]
```

Available protocols

HTTP, HTTPS

Allowed values

size

This integer specifies the maximum size of the file where web request information is written. The default value is in bytes, but you can use k, m, or g to specify units of kilobytes, megabytes, or gigabytes, respectively. If LogMaxSize is zero, the message log file size is unlimited.

Default

0

Remarks

When the message log file reaches the stated size, it is renamed and another log file is created.

17.2.14 LogOptions (LOPT) Communication Parameter

Controls which categories of messages are logged.

Syntax

```
{ LogOptions | LOPT }={ NONE | OK | INFO | ERRORS | ALL | <status-codes> |
REQHDRS | RESHDRS | HEADERS }
```

Usage

HTTP, HTTPS

Allowed Values

NONE, OK, INFO, ERRORS, ALL, <status-codes>, REQHDRS, RESHDRS, HEADERS

Default

ALL

Remarks

The values available include keywords that select particular types of messages, and HTTP status codes. Multiple values may be specified, separated by commas.

The following keywords control which categories of messages are logged:

- Log nothing.
- Log requests that complete successfully (20x HTTP status codes).
- Log requests that return over or not modified status codes (30x HTTP status codes).
- Log all errors (40x and 50x HTTP status codes)
- Log all requests.

The following common HTTP status codes are also available. They can be used to log requests that return particular status codes:

- OK
- Bad request
- Unauthorized
- Forbidden
- Not found
- Request timeout
- Not implemented
- Service unavailable

In addition, you can use these keywords to obtain more information about the logged messages:

- When logging requests, also write request headers to the log file.
- When logging requests, also write response headers to the log file.
- When logging requests, also write both request and response headers to the log file (same as REQHDRS,RESHDRS).

17.2.15 MaxConnections (MAXCONN) Communication Parameter

Specifies the number of simultaneous connections accepted by the server.

Syntax

```
{ MaxConnections | MAXCONN }=<INTEGER>
```

Usage

HTTP, HTTPS

Allowed Values

Number

Default

Number of licensed connections

Remarks

The value 0 indicates no limit.

17.2.16 MaxRequestSize (MAXSIZE) protocol option

Specifies the size of the largest request the database server can accept.

Syntax

```
{ MaxRequestSize | MAXSIZE }=<size>[ k | m | g ]
```

Available protocols

HTTP, HTTPS

Allowed values

size

This integer specifies the size of the largest request the database server can accept. The default value is in bytes, but you can use k, m, or g to specify units of kilobytes, megabytes, or gigabytes, respectively. The value 0 disables this limit, but should be used with extreme caution. Without this limit, a rogue client could overload the database server or cause it to run out of memory.

Default

100k

Remarks

If the size of a request exceeds this limit, the connection is closed and the client returns a response indicating that the request is too large. This value limits only the size of the request, not that of the response.

Example

The following command instructs the database server to accept requests up to 150000 bytes in size:

```
start_iq -xs http(MaxRequestSize=150000)
```

17.2.17 MyIP (ME) Communication Parameter

Controls whether the client attempts to determine addressing information.

Syntax

```
{ MyIP | ME }=<STRING>
```

Usage

TCP/IP, HTTP, HTTPS

Values

String

Remarks

The MyIP (ME) parameter is provided for computers with more than one network adapter.

Each adapter has an IP address. By default, SAP IQ uses the first network card it finds. If you want your database server to use more than one network card, specify the address of each card in the MyIP (ME) parameter.

If you supply the keyword NONE as the IP number, no attempt is made to determine the addressing information. NONE is intended primarily for clients on operating systems where this operation is expensive, such as machines that have multiple network cards or remote access (RAS) software and a network card. It is not intended for use on the server.

On Windows platforms, you can use this option multiple times for machines with multiple IP addresses.

Separate multiple IP addresses with commas. You can optionally append a port number to the IP address, separated by a colon.

When specifying an IPv6 address on a Windows platform, use the interface identifier. UNIX platforms support both interface identifiers and interface names in IPv6 addresses. The interface identifier is required on Linux (kernel 2.6.13 and later).

Examples

- This Windows command line (entered all on one line) instructs the server to use two network cards, one with a specified port number:

```
start_iq -x tcpip(MyIP=192.75.209.12:2367,192.75.209.32)
c:\SAP\IQ-16_1\demo\iqdemo.db
```

- This command line (entered all on one line) instructs the server to use an IPv6 network card:

```
start_iq -x tcpip(MyIP=fe80::5445:5245:444f)
"c:\SAP\IQ-16_1\demo\iqdemo.db"
```

- This connection string fragment instructs the client to make no attempt to determine addressing information:

```
...CommLinks= tcpip(MyIP=NONE)...
```

17.2.18 PreFetchOnOpen Communication Parameter

Sends a prefetch request with a cursor open request, thereby eliminating a network request to fetch rows each time a cursor is opened.

Syntax

```
PreFetchOnOpen={ YES | NO }
```

Usage

ODBC

Values

YES, NO

Default

NO

Remarks

Columns must already be bound for the prefetch to occur on the open. Rebinding columns between the cursor open and the first fetch when using PreFetchOnOpen reduces performance.

Calling ODBC's SQLExecute or SQLExecDirect on a query or stored procedure, which returns a result set causes a cursor open.

Enabling this option can improve performance in the following conditions:

- If your network exhibits poor latency
- If your application sends many cursor open and close requests

17.2.19 ReceiveBufferSize (RCVBUFSZ) Communication Parameter

Sets the size for a buffer used by the TCP/IP protocol stack. You may want to increase the value if LOB performance over the network is important.

Syntax

```
{ ReceiveBufferSize | RCVBUFSZ }=<INTEGER> [ k | m | g ]
```

Usage

TCP/IP

Allowed Values

Integer

The maximum allowed value is 1048576 bytes (1 MB)

Default

Machine-dependent

Remarks

Use `k`, `m`, or `g` to specify units of kilobytes, megabytes, or gigabytes, respectively.

17.2.20 SendBufferSize (SNDBUFSZ) Communication Parameter

Sets the size for a buffer used by the TCP/IP protocol stack.

Syntax

```
{ SendBufferSize | SNDBUFSZ }=<INTEGER> [ k | m | g ]
```

Usage

TCP/IP

Allowed Values

Integer

The maximum allowed value is 1048576 bytes (1 MB)

Default

Computer-dependent

Remarks

The default value is in bytes, but you can use `k`, `m`, or `g` to specify kilobytes, megabytes, or gigabytes, respectively. You may want to increase the value if LOB performance over the network is important.

17.2.21 ServerPort (PORT) Communication Parameter

In the case of the database server, the `ServerPort` option designates the port number on which to communicate using TCP/IP. In a data source, the `ServerPort` option informs the client of the port or ports on which database servers are listening for TCP/IP communication.

Syntax

```
{ ServerPort | PORT }=<INTEGER>
```

Usage

TCP/IP (all platforms), HTTP, HTTPS

Allowed Values

Integer

Default

The default value for TCP/IP is 2638. The default value for HTTP is 80. The default value for HTTPS is 443.

Description

The Internet Assigned Numbers Authority has assigned the SAP IQ database server port number 2638 to use for TCP/IP communications. However, applications are not disallowed from using this reserved port, and this may result in an addressing collision between the database server and another application.

The client broadcasts to every port that is specified on the `ServerPort` parameter to find the server. If `ServerPort` is unspecified, the default port 2638 is used.

The database server uses the port number specified in `ServerPort` for TCP/IP connections. Since the default port (2638) is used when you do not specify `ServerPort`, applications can connect to the database server without specifying a port number. An exception is the HP-UX operating system, on which the server does not listen on port 2638 if it is started on another port.

By default, the database server listens on the standard HTTP and HTTPS ports of 80 and 443, respectively.

Example

1. On Windows, start an SAP IQ network server:

   ```
   start_iq -x tcpip c:\SAP\IQ-16_1\demo\iqdemo.db
   ```

   Port number 2638 is now taken.
2. Attempt to start another database server:

   ```
   start_iq -x tcpip c:\SAP\IQ-16_1\demo\iqdemo2.db
   ```

   The default port being allocated, the server starts on another port.
3. Start another database server on port 2639:

   ```
   start_iq -x "tcpip(ServerPort=2639)" c:\SAP\IQ-16_1\demo\iqdemo3.db
   ```

   This succeeds as long as 2639 is not a reserved port and no other application has allocated it.
4. If another web server on your machine is already using port 80 or you do not have permission to start a server on such a low port number, start a server that listens on an alternate port, such as 8080:

   ```
   start_iq -xs http{port=8080} -n server3 web.db
   ```

17.2.22 Sessions Communication Parameter

Sets the maximum number of clients that can communicate with the server at one time through a single LAN adapter.

Syntax

```
Sessions=<INTEGER>
```

Usage

NetBIOS. Server side only.

Default

OS-specific. On Windows, the default is 16.

Remarks

The default setting is operating-system specific. The value is an integer, with maximum value 254.

NetBIOS network software has a limit to the number of commands allowed per machine. SAP IQ uses these NetBIOS commands, and disallows further connections if the system has no more commands available, even if the number of NetBIOS commands is fewer than the value of the Sessions parameter.

Example

The following statement starts a server with a database named `iqdemo`, allowing 200 NetBIOS connections:

```
start_iq -x netbios(sessions=200) iqdemo.db
```

17.2.23 TDS Communication Parameter

Controls whether the server allows TDS connections.

Syntax

```
TDS={ YES | NO }
```

Usage

TCP/IP, NamedPipes (server side only)

Allowed Values

YES, NO

Default

YES

Remarks

To disallow TDS connections to a database server, set TDS to NO. To ensure that only encrypted connections are made to your server, these port options are the only way to disallow TDS connections.

Example

The following command starts a database server that uses the TCP/IP protocol, but disallows connections from Open Client or jConnect applications:

```
start_iq -x tcpip(TDS=NO) ...
```

17.2.24 Timeout (TO) Communication Parameter

Specifies the length of time, in seconds, to wait for a response when establishing communications and when disconnecting.

Syntax

```
{ Timeout | TO }=<INTEGER>
```

Usage

TCP/IP (all platforms), HTTP, HTTPS

Allowed Values

Integer, in seconds

Maximum Allowed Value

3600 seconds

Default

5

Remarks

You may want to try longer times if you are having trouble establishing TCP/IP communications.

In HTTP or HTTPS applications, this parameter specifies the maximum idle time permitted when receiving a request. If this limit is reached, the connection is closed and a `408 REQUEST TIMEOUT` is returned to the client. The value 0 disables idle timeout, and should be used with extreme caution. Without this limit, a rogue client might consume the server's resources and prevent other clients from connecting.

Example

The following data source fragment starts a TCP/IP communications link only, with a timeout period of 20 seconds:

```
...
CommLinks=tcpip(TO=20)
...
```

17.2.25 VerifyServerName (VERIFY) Communication Parameter

Specifies whether the server name is verified when connecting to this host.

Syntax

```
{ VerifyServerName | VERIFY }={ YES | NO }
```

Usage

TCP/IP (Client side only)

Allowed Values

YES, NO

Default

YES

Remarks

Normally, you should not set this option. It is used only for connecting to multiplex secondary servers to balance query loads among these servers.

When connecting over TCP using the `DoBroadcast=NONE` parameter, the client makes a TCP connection, then verifies that the name of the server found is the same as the one it is looking for. Specifying `VerifyServerName=NO` skips the verification of the server name. This allows IQ clients to connect to an SAP IQ server if they know only an IP address/port.

The server name must still be specified in the connection string, but it is ignored. The `VerifyServerName` communication parameter is used only if `DoBroadcast=NONE` is specified.

When used as shown in the example, setting this option to NO lets you specify a connection to a particular IP address and port number. The IP address and port number are for a load balancing machine that acts as a gateway between the IQ client and the IQ server.

Example

To use this option, create a new ODBC DSN in the ODBC Administrator on the client machine, and specify parameters as follows:

- On the Database tab, specify a generic server name for connecting to all of the secondary servers, for example, `qserv`. A server name is required, but ignored because of parameters in the Network tab.
- On the Network tab, select the TCP/IP check box and type in the text box:

  ```
  host=<hostname>;port=<port#>;DOBROADCAST=NONE;VERIFY=NO
  ```

  For example:

  ```
  host=<hostname>;port=2222;DOBROADCAST=NONE;VERIFY=NO
  ```

When an IQ client connects to this DSN, the load balancer dispatches the connection to a particular secondary server based on the workload of the machine.

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 : 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 : You are leaving the documentation for that particular SAP product or service and are entering a 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.

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.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

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.

www.sap.com/contactsap

© 2020 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.

THE BEST RUN SAP