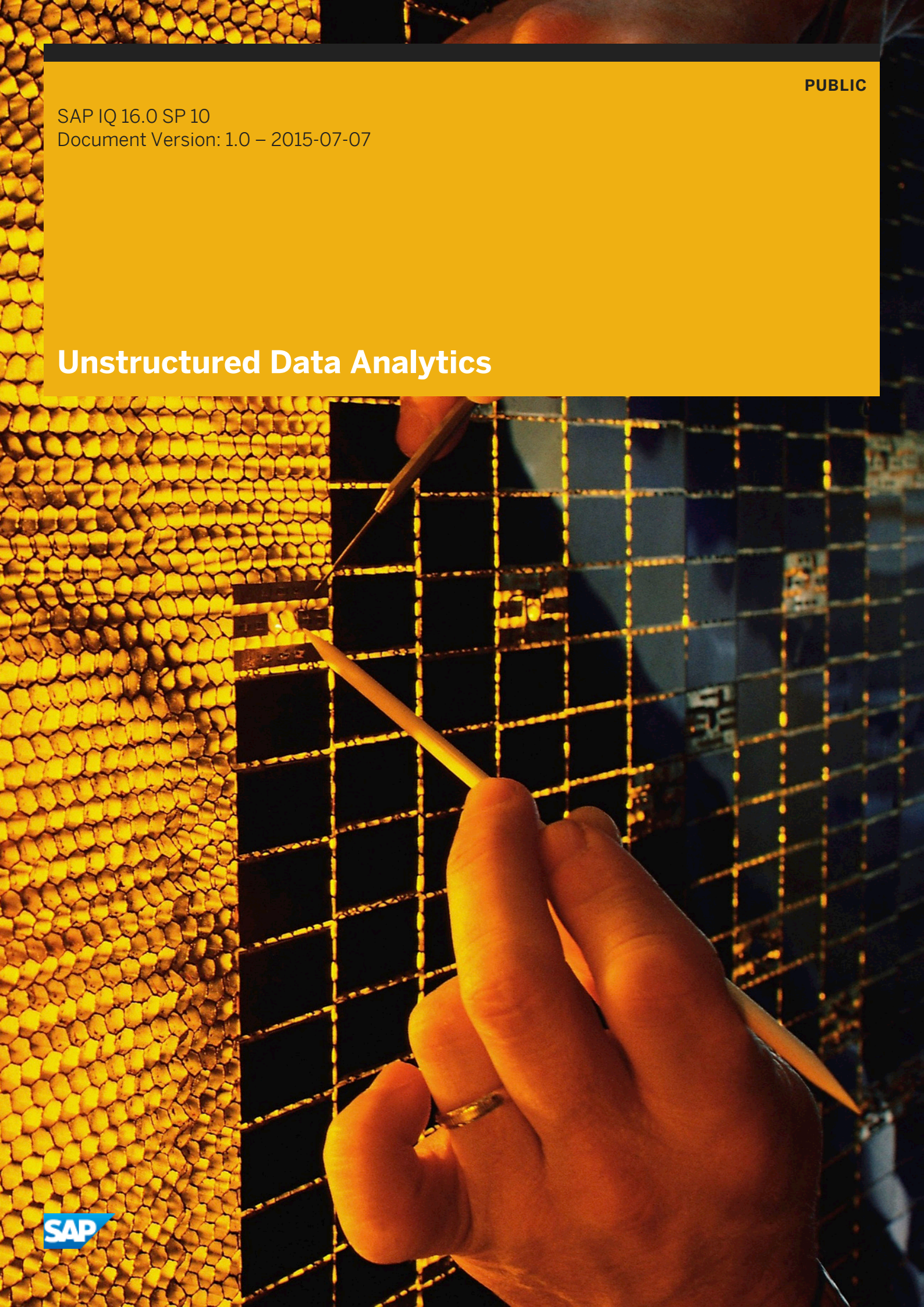
PUBLIC

SAP IQ 16.0 SP 10
Document Version: 1.0 – 2015-07-07

Unstructured Data Analytics



Content

1 Introduction to Unstructured Data Analytics. 5
1.1 Audience. 5
1.2 The Unstructured Data Analytics Option. 5
Full Text Searching. 6
1.3 Compatibility. 6
1.4 Conformance to Standards. 6

2 TEXT Indexes and Text Configuration Objects. 7
2.1 TEXT Indexes. 7
Comparison of WD and TEXT Indexes. 7
Creating a TEXT Index Using Interactive SQL. 9
Guidelines for TEXT Index Size Estimation. 9
TEXT Index Restrictions. 10
Displaying a List of TEXT Indexes Using Interactive SQL. 10
Editing a TEXT Index Using Interactive SQL. 11
Modifying the TEXT Index Location Using Interactive SQL. 11
Dropping a TEXT Index Using Interactive SQL. 12
TEXT Index Refresh. 12
TEXT_DELETE_METHOD Database Option. 12
2.2 NGRAM TEXT Index. 14
2.3 Text Configuration Objects. 14
Default Text Configuration Objects. 15
Creating a Text Configuration Using Interactive SQL. 15
Text Configuration Object Settings. 16
Displaying a List of Text Configurations Using Interactive SQL. 19
Altering a Text Configuration Using Interactive SQL. 19
Modifying the Stoplist Using Interactive SQL. 20
Dropping a Text Configuration Using Interactive SQL. 21
Text Configuration Object Examples. 21
MAX_PREFIX_PER_CONTAINS_PHRASE Database Option. 24

3 External Libraries. 26
3.1 Pre-Filter and Term-Breaker External Libraries. 26
External Library Restrictions. 27
3.2 External Libraries on Multiplex Servers. 27
3.3 Enable and Disable External Libraries on Startup. 27
3.4 Unload External Libraries. 28

4 Unstructured Data Queries. 29
4.1 Full Text Search. 29
Types of Full Text Searches. 29
4.2 NGRAM TEXT Index Searches. 53
Fuzzy Search Over a TEXT Index. 53
Non-fuzzy Search Over a TEXT Index. 55
4.3 Queries on LONG BINARY Columns. 56
4.4 Queries on LONG VARCHAR Columns. 57
CONTAINS Predicate Support. 57
4.5 Performance Monitoring of LONG BINARY and LONG VARCHAR Columns. 57

5 Stored Procedure Support. 58
5.1 Term Management in a TEXT Index. 58
sa_char_terms System Procedure. 58
sa_nchar_terms System Procedure. 59
sa_text_index_stats System Procedure. 60
sa_text_index_vocab System Procedure. 62
5.2 External Library Identification. 63
sa_external_library_unload System Procedure. 63
sa_list_external_library System Procedure. 64
5.3 Large Object Data Compression. 65
sp_iqsetcompression Procedure. 65
sp_iqshowcompression Procedure. 67
5.4 Information About Large Object Columns. 68
Size of a LONG BINARY Column. 68
Size of a LONG VARCHAR Column. 68

6 Large Object Data Load and Unload. 70
6.1 Large Object Data Exports. 70
BFILE Function. 70
6.2 Large Object Data Loads. 72
Extended LOAD TABLE Syntax. 72
Large Object Data Load Example. 73
Control of Load Errors. 74
Load of Large Object Data with Trailing Blanks. 74
Load of Large Object Data with Quotes. 74
Truncation of Partial Multibyte Character Data. 74
Load Support of Large Object Variables. 75

7 Large Object Data Types. 76
7.1 Large Object Data Types LONG BINARY and BLOB. 76
LONG BINARY Data Type Conversion. 76

7.2 Large Object Data Types LONG VARCHAR and CLOB. 77
LONG VARCHAR Data Type Conversion. 78
7.3 Large Object Variables. 78
Large Object Variable Data Type Conversion. 79
7.4 Index Support of Large Object Columns. 80
TEXT Index Support of Large Object Columns. 80
WD Index Support of LONG VARCHAR (CLOB) Columns. 81

8 SQL Statement Support. 82
8.1 ALTER TEXT CONFIGURATION Statement. 82
8.2 ALTER TEXT INDEX Statement. 85
8.3 CREATE TEXT CONFIGURATION Statement. 86
8.4 CREATE TEXT INDEX Statement. 88
8.5 DROP TEXT CONFIGURATION Statement. 90
8.6 DROP TEXT INDEX Statement. 91

9 Function Support. 93
9.1 Summary of Function Support of Large Object Data. 93
9.2 BIT_LENGTH Function. 94
9.3 BYTE_LENGTH Function [String]. 94
9.4 BYTE_LENGTH64 Function [String]. 96
9.5 BYTE_SUBSTR64 and BYTE_SUBSTR Functions [String]. 96
9.6 CHAR_LENGTH Function [String]. 97
9.7 CHAR_LENGTH64 Function [String]. 98
9.8 CHARINDEX Function [String]. 98
9.9 LOCATE Function [String]. 99
9.10 OCTET_LENGTH Function [String]. 100
9.11 PATINDEX Function [String]. 101
9.12 SUBSTRING Function [String]. 102
ANSI_SUBSTRING Option [TSQL]. 103
9.13 SUBSTRING64 Function [String]. 104
9.14 Aggregate Function Support of Large Object Columns. 105
9.15 User-Defined Function Support of Large Object Columns. 105

1 Introduction to Unstructured Data Analytics

Learn about unstructured data analytics in SAP® IQ and about large object data compatibility and conformance.

1.1 Audience

This guide is for users who require reference material for working with unstructured data in SAP IQ.

Learn about available syntax, parameters, functions, stored procedures, indexes, and options related to unstructured data analytics features. Use this guide as a reference to understand storage and retrieval of unstructured data within the database.

1.2 The Unstructured Data Analytics Option

The Unstructured Data Analytics Option extends the capabilities of SAP IQ to allow storage, retrieval, and full text searching of binary large objects (BLOBs) and character large objects (CLOBs) within the database.

> **Note**
>
> Users must be specifically licensed to use the Unstructured Data Analytics functionality described in this product documentation.

As data volumes increase, the need to store large object (LOB) data in a relational database also increases. LOB data may be either:

- Unstructured – the database simply stores and retrieves the data, or
- Semistructured (for example, text) – the database supports the data structure and provides supporting functions (for example, string functions).

Typical LOB data sources include images, maps, documents (for example, PDF files, word processing files, and presentations), audio, video, and XML files. SAP IQ can manage individual LOB objects containing gigabytes (GB), terabytes (TB), or even petabytes (PB) of data.

By allowing relational and unstructured data in the same location, SAP IQ lets organizations access both types of data using the same application and the same interface. The full text search capability of SAP IQ supports text archival applications (text analytics) in handling unstructured and semistructured data.

1.2.1 Full Text Searching

Full text searching uses `TEXT` indexes to search for terms and phrases in a database without having to scan table rows.

A `TEXT` index stores positional information for terms in the indexed column. Text configuration objects control the terms that are placed in a `TEXT` index when it is built or refreshed, and how a full text query is interpreted.

Using a `TEXT` index to find rows that contain a term or phrase is generally faster than scanning every row in the table.

1.3 Compatibility

SAP® SQL Anywhere® (SA) and SAP® Adaptive Server® Enterprise (SAP ASE) (ASE) store large text and binary objects.

SAP SQL Anywhere can store large objects (up to a 2GB maximum length) in columns of data type `LONG VARCHAR` or `LONG BINARY`. The support of these data types by SAP SQL Anywhere is SQL/2003 compliant. SAP SQL Anywhere does not support the `BYTE_LENGTH64`, `BYTE_SUBSTR64`, `BFILE`, `BIT_LENGTH`, `OCTET_LENGTH`, `CHAR_LENGTH64`, and `SUBSTRING64` functions.

SAP ASE can store large text objects (up to a 2GB maximum length) and large binary objects (up to a 2GB maximum length) in columns of data type `TEXT` or `IMAGE`, respectively. The support of these data types by SAP ASE is an ANSI SQL Transact-SQL® extension.

A `LONG BINARY` column of a proxy table maps to a `VARBINARY(max)` column in a Microsoft SQL Server table.

1.4 Conformance to Standards

SAP IQ `LONG BINARY` and `LONG VARCHAR` functionality conforms to the Core level of the ISO/ANSI SQL standard.

2 TEXT Indexes and Text Configuration Objects

Learn about working with TEXT indexes and text configuration objects.

A TEXT index stores positional information for terms in an indexed column. TEXT indexes are created using settings stored in a text configuration object. A text configuration object controls characteristics of TEXT index data, such as terms to ignore, and the minimum and maximum length of terms to include in the index.

2.1 TEXT Indexes

In a full text search, a TEXT index is searched, rather than table rows.

Before you can perform a full text search, you must create a TEXT index on the columns you want to search. A TEXT index stores positional information for terms in the indexed columns. Queries that use TEXT indexes are generally faster than those that must scan all the values in the table.

When you create a TEXT index, you can specify which text configuration object to use when creating and refreshing the TEXT index. A text configuration object contains settings that affect how an index is built. If you do not specify a text configuration object, the database server uses a default configuration object.

You can create TEXT indexes on these types of columns: CHAR, VARCHAR, and LONG VARCHAR, as well as BINARY, VARBINARY, and LONG BINARY. BINARY, VARBINARY, and LONG BINARY columns require that the TEXT index use a text configuration with an external prefilter library.

2.1.1 Comparison of WD and TEXT Indexes

A comparison of WD and TEXT indexes in terms of syntax and capability.

Table 1: WD versus TEXT Index

Feature	Supported by WD index?	Supported by TEXT index?
Conjunction of terms	Yes, expressed in the form: `tbl.col CONTAINS('great','white','whale')`	Yes, expressed in the form: `CONTAINS(tbl.col,'great white whale')`

Feature	Supported by WD index?	Supported by TEXT index?
General boolean expressions	Yes, expressed in the form: `tbl.col CONTAINS ('great') AND ( tbl.col CONTAINS('white) OR tbl.col CONTAINS('whale') AND NOT tbl.col CONTAINS('ship'))`	Yes, expressed in the form: `CONTAINS(tbl.col, 'great AND ( white OR whale AND NOT ship )')`
Search for terms matching prefix	No	Yes, for example: `CONTAINS (tbl.col,'whale*')`
Acceleration of `LIKE` predicates	Yes, for example: `tbl.col LIKE 'whale%'`	Yes, if the TEXT index is an NGRAM TEXT index without an external document pre-filter. For example: `c LIKE '%apple%fruit'`

Searches for terms in proximity	No	Yes, for example:
		`CONTAINS(tbl.col, 'white BEFORE whale')`
		`CONTAINS(tbl.col, 'whale NEAR white')`
		`CONTAINS(tbl.col, ' "white whale" ')`

Ordering of results based on search scoring	No	Yes

In `TEXT` index, searching for terms matching a prefix and searching for a `LIKE` expression have different semantics and may return very different results depending on the text configuration. The specification of minimum length, maximum length and a stoplist will govern the prefix processing but does not affect `LIKE` semantics.

> **Note**
>
> Meaning of boolean expressions will differ between `WD` index and `TEXT` index when term dropping occurs, because the effect of dropped terms in `TEXT` index processing has no equivalent in the `WD` index.

2.1.2 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
```

2.1.3 Guidelines for TEXT Index Size Estimation

Formula estimates `TEXT` index main store size.

```
<Number of bytes = (15+L)*U + U*PAGESIZE * R + T>
```

where:

- L = average term length for the vocabulary
- U = number of unique terms in the vocabulary
- R = number of millions of documents
- T = total number of all terms in all documents

The temporary space required in bytes for the `TEXT` index is `<(M+20)* T>`, where:

- M = the maximum term length for the text configuration in bytes

Note

The temporary space required is subject to compressibility of the sort data.

2.1.4 TEXT Index Restrictions

Text configuration objects and `TEXT` indexes have limitations by design.

- SAP IQ does not provide support for `TEXT` indexes spanning multiple columns.
- `TEXT` index manual refresh or automatic refresh options are not supported.
- `sp_iqrebuildindex` cannot be used to build `TEXT` indexes.
- You cannot create `TEXT` indexes within `BEGIN PARALLEL IQ…END PARALLEL IQ`.
- `NGRAM` term breaker is built on `TEXT` indexes, so use text configuration object settings to define whether to use an `NGRAM` or `GENERIC TEXT` index.
- `NGRAM TEXT` index search is mainly useful when words are misspelled. SAP IQ does not support searches like synonyms and antonyms.

2.1.5 Displaying a List of TEXT Indexes Using Interactive SQL

View a list of all the `TEXT` indexes in the database.

Procedure

1. Connect to the database as a user with any of the following system privileges:
   - CREATE ANY INDEX
   - ALTER ANY INDEX
   - DROP ANY INDEX
   - CREATE ANY OBJECT
   - ALTER ANY OBJECT
   - DROP ANY OBJECT
   - MANAGE ANY DBSPACE
2. Execute a `SELECT` statement.

Example

To list all `TEXT` indexes:

```
SELECT * FROM sp_iqindex() WHERE index_type = 'TEXT';
```

To list all `TEXT` indexes, including those on catalog tables:

```
SELECT index_name, table_name, name FROM SYSIDX, SYSTEXTIDX, SYSTABLE, SYSUSERS
WHERE SYSIDX.object_id=SYSTEXTIDX.index_id
AND SYSIDX.table_id=SYSTABLE.table_id
AND SYSTABLE.creator=SYSUSERS.uid;
```

2.1.6 Editing a TEXT Index Using Interactive SQL

Change the settings for the `TEXT` index, including the dbspace and `TEXT` index name.

Procedure

1. Connect to the database as a user with ALTER ANY INDEX or ALTER ANY OBJECT system privilege or as the owner of the underlying table.
2. Execute an `ALTER TEXT INDEX` statement.

Example

To rename the `TEXT` index `myTxtIdx` to `MyTextIndex`:

```
ALTER TEXT INDEX MyTxtIdx
    ON Customers
RENAME AS MyTextIndex;
```

2.1.7 Modifying the TEXT Index Location Using Interactive SQL

Change the dbspace where the `TEXT` index is stored.

Procedure

1. Connect to the database as a user with MANAGE ANY DBSPACE system privilege.
2. Execute an `ALTER TEXT INDEX` statement with the `MOVE TO` clause.

Example

To move the `TEXT` index `MyTextIndex` to a dbspace named `tispace`:

```
ALTER TEXT INDEX MyTextIndex ON
    GROUPO.customers MOVE TO tispace;
```

2.1.8 Dropping a TEXT Index Using Interactive SQL

Drop a `TEXT` index from the database.

Procedure

1. Connect to the database as a user with DROP ANY INDEX or DROP ANY OBJECT system privilege or as the owner of the underlying table.
2. Execute a `DROP TEXT INDEX` statement.

Example

To drop the `MyTextIndex` `TEXT` index:

```
DROP TEXT INDEX MyTextIndex ON Customers;
```

2.1.9 TEXT Index Refresh

The only supported refresh type for `TEXT` indexes on SAP IQ tables is Immediate Refresh, which occurs when data in the underlying table changes.

Immediate-refresh `TEXT` indexes on tables support isolation level 3. They are populated at creation time and every time the data in the column is changed using an `INSERT`, `UPDATE`, or `DELETE` statement. An exclusive lock is held on the table during the initial refresh.

2.1.10 TEXT_DELETE_METHOD Database Option

Specifies the algorithm used during a delete in a `TEXT` index.

Allowed Values

0 – 2

0 – the delete method is selected by the cost model.

1 – forces small method for deletion. Small method is useful when the number of rows being deleted is a very small percentage of the total number of rows in the table. Small delete can randomly access the index, causing cache thrashing with large data sets.

2 – forces large method for deletion. This algorithm scans the entire index searching for rows to delete. Large method is useful when the number of rows being deleted is a high percentage of the total number of rows in the table.

Default

0

Scope

Requires the SET ANY PUBLIC OPTION system privilege to set this option for PUBLIC or for other user or role. Can be set temporary only for the PUBLIC role. Takes effect immediately.

Description

`TEXT_DELETE_METHOD` specifies the algorithm used during a delete operation in a `TEXT` index. When this option is not set or is set to 0, the delete method is selected by the cost model. The cost model considers the CPU-related costs as well as I/O-related costs in selecting the appropriate delete algorithm. The cost model takes into account:

- Rows deleted
- Index size
- Width of index data type
- Cardinality of index data
- Available temporary cache
- Machine-related I/O and CPU characteristics
- Available CPUs and threads

Example

To force the large method for deletion from a `TEXT` index:

```
SET TEMPORARY OPTION TEXT_DELETE_METHOD = 2
```

2.2 NGRAM TEXT Index

NGRAM TEXT index stores the text in the column by breaking the text into n-grams of text value N, where N is the value given by a user.

You can perform a search over an `NGRAM TEXT` index by matching the n-grams of the text value in the `CONTAINS` clause of the query against the stored n-grams in the index.

`NGRAM TEXT` index accommodates fuzzy searching capability over the text for both European and non-European languages.

> **Note**
>
> `NGRAM TEXT` index search is mainly useful when words are misspelled. SAP IQ does not support searches like synonyms and antonyms.

`NGRAM` term breaker is built on `TEXT` indexes, so use text configuration object settings to define whether to use an `NGRAM` or `GENERIC TEXT` index.

2.3 Text Configuration Objects

Text configuration objects control the terms that are placed in a `TEXT` index when it is built or refreshed, and how a full text query is interpreted.

When the database server creates or refreshes a `TEXT` index, it uses the settings for the text configuration object specified when the `TEXT` index was created. If a text configuration object is not specified, the database server chooses one of the default text configuration objects, based on the type of data in the columns being indexed. In an SAP IQ database, the `default_char` text configuration object is always used.

Text configuration objects specify which prefilter library and which term breaker are used to generate terms from the documents to be indexed. They specify the minimum and maximum length of terms to be stored within the `TEXT` index, along with the list of terms that should not be included. Text configuration objects consist of these parameters:

- Document pre-filter – removes unnecessary information, such as formatting and images, from the document. The filtered document is then picked up by other modules for further processing. The document pre-filter is provided by a third-party vendor.
- Document term-breaker – breaks the incoming byte stream into terms separated by term separators or according to specified rules. The document term-breaker is provided by the server or a third-party vendor.
- Stoplist processor – specifies the list of terms to be ignored while building the `TEXT` index.

2.3.1 Default Text Configuration Objects

SAP IQ provides default text configuration objects.

The default text configuration object `default_char` is used with non-`NCHAR` data. This configuration is created the first time you create a text configuration object or `TEXT` index.

The text configuration object `default_nchar` is supported for use with `NCHAR` for `TEXT` indexes on `IN SYSTEM` tables; you cannot use `default_nchar` text configuration for `TEXT` indexes on tables.

The table "Default text configuration settings" shows the default settings for `default_char` and `default_nchar`, which are best suited for most character-based languages. We strongly suggest that you do not change the settings in the default text configuration objects.

Table 2: Default text configuration settings

Setting	Installed value
TERM BREAKER	GENERIC
MINIMUM TERM LENGTH	1
MAXIMUM TERM LENGTH	20
STOPLIST	(empty)

If you delete a default text configuration object, it is automatically re-created with default values the next time you create a `TEXT` index or text configuration object.

2.3.2 Creating a Text Configuration Using Interactive SQL

Create a text configuration to specify how `TEXT` indexes dependent on the text configuration process handle terms within the data.

Procedure

1. Connect to the database as a user with CREATE TEXT CONFIGURATION or CREATE ANY TEXT CONFIGURATION or CREATE ANY OBJECT system privilege.
2. Execute a `CREATE TEXT CONFIGURATION` statement.

Example

To create a text configuration object called `myTxtConfig` using the `default_char` text configuration object as a template:

```
CREATE TEXT CONFIGURATION myTxtConfig FROM default_char;
```

2.3.3 Text Configuration Object Settings

Learn about text configuration object settings, how they affect what is indexed, and how a full text search query is interpreted.

Related Information

Text Configuration Object Setting Interpretations [page 21]

2.3.3.1 Term Breaker Algorithm (TERM BREAKER)

The `TERM BREAKER` setting specifies the algorithm to use for breaking strings into terms.

SAP IQ `GENERIC` (the default) or `NGRAM` for storing terms.

> **Note**
>
> `NGRAM` term breakers store n-grams. An n-gram is a group of characters of length `<n>` where `<n>` is the value of `MAXIMUM TERM LENGTH`.

Regardless of the term breaker you specify, the database server records in the `TEXT` index the original positional information for terms when they are inserted into the `TEXT` index. In the case of n-grams, the positional information of the n-grams is stored, not the positional information for the original terms.

Table 3: TERM BREAKER impact

To TEXT index	To query terms
GENERIC TEXT index when building a `GENERIC TEXT` index (the default), groups of alphanumeric characters appearing between non-alphanumeric characters are processed as terms by the database server. After the terms have been defined, terms that exceed the term length settings, and terms found in the stoplist, are counted but not inserted in the `TEXT` index. Performance on `GENERIC TEXT` indexes can be faster than `NGRAM TEXT` indexes. However, you cannot perform fuzzy searches on `GENERIC TEXT` indexes.	**GENERIC TEXT index** when querying a `GENERIC TEXT` index, terms in the query string are processed in the same manner as if they were being indexed. Matching is performed by comparing query terms to terms in the `TEXT` index.

To TEXT index	To query terms
NGRAM TEXT index when building an `NGRAM TEXT` index, the database server treats as a term any group of alphanumeric characters between non-alphanumeric characters. Once the terms are defined, the database server breaks the terms into n-grams. In doing so, terms shorter than n, and n-grams that are in the stoplist, are discarded. For example, for an `NGRAM TEXT` index with `MAXIMUM TERM LENGTH` 3, the string 'my red table' is represented in the `TEXT` index as these n-grams: red tab abl ble.	**NGRAM TEXT index** when querying an `NGRAM TEXT` index, terms in the query string are processed in the same manner as if they were being indexed. Matching is performed by comparing n-grams from the query terms to n-grams from the indexed terms.

2.3.3.2 Minimum Term Length Setting (MINIMUM TERM LENGTH)

The `MINIMUM TERM LENGTH` setting specifies the minimum length, in characters, for terms inserted in the index or searched for in a full text query.

`MINIMUM TERM LENGTH` is not relevant for `NGRAM TEXT` indexes.

`MINIMUM TERM LENGTH` has special implications on prefix searching. The value of `MINIMUM TERM LENGTH` must be greater than 0. If you set it higher than `MAXIMUM TERM LENGTH`, then `MAXIMUM TERM LENGTH` is automatically adjusted to be equal to `MINIMUM TERM LENGTH`.

The default for `MINIMUM TERM LENGTH` is taken from the setting in the default text configuration object, which is typically 1.

Table 4: MINIMUM TERM LENGTH impact

To TEXT index	To query terms
GENERIC TEXT index for `GENERIC TEXT` indexes, the `TEXT` index will not contain words shorter than `MINIMUM TERM LENGTH`.	**GENERIC TEXT index** when querying a `GENERIC TEXT` index, query terms shorter than `MINIMUM TERM LENGTH` are ignored because they cannot exist in the `TEXT` index.
NGRAM TEXT index for `NGRAM TEXT` indexes, this setting is ignored.	**NGRAM TEXT index** the `MINIMUM TERM LENGTH` setting has no impact on full text queries on `NGRAM TEXT` indexes.

2.3.3.3 Maximum Term Length Setting (MAXIMUM TERM LENGTH)

The `MAXIMUM TERM LENGTH` setting specifies the maximum length, in characters, for terms inserted in the index or searched for in a full text query.

The `MAXIMUM TERM LENGTH` setting is used differently, depending on the term breaker algorithm. The value of `MAXIMUM TERM LENGTH` must be less than or equal to 60. If you set `MAXIMUM TERM LENGTH` lower than the `MINIMUM TERM LENGTH`, then `MINIMUM TERM LENGTH` is automatically adjusted to be equal to `MAXIMUM TERM LENGTH`.

The default for this setting is taken from the setting in the default text configuration object, which is typically 20.

Table 5: MAXIMUM TERM LENGTH impact

To TEXT index	To query terms
GENERIC TEXT index for `GENERIC TEXT` indexes, `MAXIMUM TERM LENGTH` specifies the maximum length, in characters, for terms inserted in the `TEXT` index.	**GENERIC TEXT index** for `GENERIC TEXT` indexes, query terms longer than `MAXIMUM TERM LENGTH` are ignored because they cannot exist in the `TEXT` index.
NGRAM TEXT index for `NGRAM TEXT` indexes, `MAXIMUM TERM LENGTH` determines the length of the n-grams that terms are broken into. An appropriate choice of length for `MAXIMUM TERM LENGTH` depends on the language. Typical values are 4 or 5 characters for English, and 2 or 3 characters for Chinese.	**NGRAM TEXT index** for `NGRAM TEXT` indexes, query terms are broken into n-grams of length n, where n is the same as `MAXIMUM TERM LENGTH`. The database server uses the n-grams to search the `TEXT` index. Terms shorter than `MAXIMUM TERM LENGTH` are ignored because they do not match the n-grams in the `TEXT` index.

2.3.3.4 Stoplist Setting (STOPLIST)

The stoplist setting specifies terms that are not indexed.

The default for the stoplist setting is taken from the setting in the default text configuration object, which typically has an empty stoplist.

Table 6: STOPLIST impact

To TEXT index	To query terms
GENERIC TEXT index for `GENERIC TEXT` indexes, terms that are in the stoplist are not inserted into the `TEXT` index.	**GENERIC TEXT index** for `GENERIC TEXT` indexes, query terms that are in the stoplist are ignored because they cannot exist in the `TEXT` index.
NGRAM TEXT index for `NGRAM TEXT` indexes, the `TEXT` index does not contain the n-grams formed from the terms in the stoplist.	**NGRAM TEXT index** terms in the stoplist are broken into n-grams and the n-grams are used for the stoplist. Likewise, query terms are broken into n-grams and any that match n-grams in the stoplist are dropped because they cannot exist in the `TEXT` index.

Consider carefully whether to put terms in to your stoplist. In particular, do not include words that have non-alphanumeric characters in them such as apostrophes or dashes. These characters act as term breakers. For example, the word you'll (which must be specified as 'you''ll') is broken into you and ll and stored in the stoplist as these two terms. Subsequent full text searches for 'you' or 'they'll' are negatively impacted.

Stoplists in `NGRAM TEXT` indexes can cause unexpected results because the stoplist that is stored is actually in n-gram form, not the actual stoplist terms you specified. For example, in an `NGRAM TEXT` index where `MAXIMUM TERM LENGTH` is 3, if you specify `STOPLIST` 'there', these n-grams are stored as the stoplist: the her ere. This impacts the ability to query for any terms that contain the n-grams the, her, and ere.

2.3.4 Displaying a List of Text Configurations Using Interactive SQL

View a list of all the text configurations in the database.

Procedure

1. Connect to the database as a user with CREATE TEXT CONFIGURATION or CREATE ANY TEXT CONFIGURATION or CREATE ANY OBJECT system privilege.
2. Execute a `SELECT` statement.

Example

To list all text configuration objects:

```
SELECT * FROM SYSTEXTCONFIG;
```

2.3.5 Altering a Text Configuration Using Interactive SQL

Change the settings of the text configuration object, including the dbspace and permitted term lengths range.

Context

You can alter only text configuration objects that are not being used by a `TEXT` index.

Procedure

1. Connect to the database as a user with ALTER ANY TEXT CONFIGURATION or ALTER ANY OBJECT system privilege, or as the owner of the text configuration object.
2. Execute an `ALTER TEXT CONFIGURATION` statement.

Example

To alter the minimum term length for the `myTxtConfig` text configuration object:

```
ALTER TEXT CONFIGURATION myTxtConfig
    MINIMUM TERM LENGTH 2;
```

2.3.6 Modifying the Stoplist Using Interactive SQL

Modify the stoplist, which contains a list of terms to ignore when building a `TEXT` index with this text configuration.

Context

You can alter only text configuration objects that are not being used by a `TEXT` index.

Procedure

1. Connect to the database as a user with ALTER ANY TEXT CONFIGURATION or ALTER ANY OBJECT system privilege, or as the owner of the text configuration object.
2. Execute an `ALTER TEXT CONFIGURATION` statement with the `STOPLIST` clause.

Example

To add a stoplist to the `myTxtConfig` configuration object:

```
ALTER TEXT CONFIGURATION myTxtConfig
    STOPLIST 'because about therefore only';
```

2.3.7 Dropping a Text Configuration Using Interactive SQL

Remove an unnecessary text configuration from the database.

Context

Only text configurations that are not being used by a `TEXT` index can be dropped.

Procedure

1. Connect to the database as a user with DROP ANY TEXT CONFIGURATION or DROP ANY OBJECT system privilege.
2. Execute a `DROP TEXT CONFIGURATION` statement.

Example

To drop the text configuration object `myTxtConfig`:

```
DROP TEXT CONFIGURATION myTxtConfig;
```

2.3.8 Text Configuration Object Examples

Review the samples to understand how text configuration settings impact the `TEXT` index, and how the index is interpreted.

2.3.8.1 Text Configuration Object Setting Interpretations

Examples that show the settings for different text configuration objects, how the settings impact what is indexed, and how a full text query string is interpreted.

All the examples use the string 'I'm not sure I understand'.

Table 7: Text configuration setting interpretations

Configuration settings	Terms that are indexed	Query interpretation
TERM BREAKER: GENERIC MINIMUM TERM LENGTH: 1 MAXIMUM TERM LENGTH: 20 STOPLIST: ''	`I m not sure I understand`	`'("I m" AND not sure) AND I AND understand'`
TERM BREAKER: GENERIC MINIMUM TERM LENGTH: 2 MAXIMUM TERM LENGTH: 20 STOPLIST: 'not and'	`sure understand`	`'understand'`
TERM BREAKER: GENERIC MINIMUM TERM LENGTH: 1 MAXIMUM TERM LENGTH: 20 STOPLIST: 'not and'	`I m sure I understand`	`'"I m" AND sure AND I AND understand'`

2.3.8.2 Text Configuration Object CONTAINS Query String Interpretations

Examples of how the settings of the text configuration object strings are interpreted for CONTAINS queries.

The parenthetical numbers in the Interpreted string column in the table "CONTAINS string interpretations" reflect the position information stored for each term. The numbers are for illustration purposes in the documentation. The actual stored terms do not include the parenthetical numbers.

> **Note**
>
> The maximum number of positions for a text document is 4294967295.

The interpretations in this table are only for `CONTAINS` queries. When data is parsed, `AND`, `NOT`, and `NEAR` are considered regular tokens; symbols like `*`, `I`, and others are dropped as they are not alphanumeric.

Table 8: CONTAINS string interpretations

Configuration settings	String	Interpreted string
TERM BREAKER: GENERIC MINIMUM TERM LENGTH: 3 MAXIMUM TERM LENGTH: 20	`'w*'`	`'"w*(1)"'`
	`'we*'`	`'"we*(1)"'`

Configuration settings	String	Interpreted string
	`'wea*'`	`'"wea*(1)"'`
	`'we* -the'`	`'"we*(1)" -"the(1)"'`
	`'for* | wonderl*'`	`'"for*(1)" | "wonderl*(1)"'`
	`'wonderlandwonderlandwonderland*'`	`''`
	`'"tr* weather"'`	`'"weather(1)"'`
	`'"tr* the weather"'`	`'"the(1) weather(2)"'`
	`'"wonderlandwonderlandwonderland* wonderland"'`	`'"wonderland(1)"'`
	`'"wonderlandwonderlandwonderland* weather"'`	`'"weather(1)"'`
	`'"the_wonderlandwonderlandwonderland* weather"'`	`'"the(1) weather(3)"'`
	`'the_wonderlandwonderlandwonderland* weather'`	`'"the(1)" & "weather(1)"'`
	`'"light_a* the end" & tunnel'`	`'"light(1) the(3) end(4)" & "tunnel(1)"'`
	`light_b* the end" & tunnel'`	`'"light(1) the(3) end(4)" & "tunnel(1)"'`
	`'"light_at_b* end"'`	`'"light(1) end(4)"'`
	`'and-te*'`	`'"and(1) te*(2)"'`
	`'a_long_and_t* & journey'`	`'"long(2) and(3) t*(4)" & "journey(1)"'`

2.3.9 MAX_PREFIX_PER_CONTAINS_PHRASE Database Option

Specifies the number of prefix terms allowed in a text search expression.

Allowed Values

0 – 300

0 – no limit for prefix terms in search phrase

300 – upper limit (this is the overall limit for total number of terms allowed in a phrase)

Default

1

Scope

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary, for an individual connection, or for the `PUBLIC` role. Takes effect immediately.

Description

`MAX_PREFIX_PER_CONTAINS_PHRASE` specifies the threshold used to disallow more than one prefix in an expression for a text search.

When this option is set to 0, any number is allowed. SAP IQ detects and reports an error, if the query has any `CONTAINS` expressions with a phrase having more prefix terms than specified by this option.

Examples

With the default `MAX_PREFIX_PER_CONTAINS_PHRASE` setting:

```
SET MAX_PREFIX_PER_CONTAINS_PHRASE = 1
```

this `CONTAINS` clause is valid:

```
SELECT ch1 FROM tab1
```

```
WHERE CONTAINS(ch1, '"concord bed* in mass"')
```

With the default `MAX_PREFIX_PER_CONTAINS_PHRASE` setting of 1, this `CONTAINS` clause returns a syntax error:

```
SELECT ch1 FROM tab1
WHERE CONTAINS (ch1, '"con* bed* in mass"')
```

When `MAX_PREFIX_PER_CONTAINS_PHRASE` is set equal to 0 (no limit) or 2, this `CONTAINS` clause is valid.

3 External Libraries

Learn about using external libraries to supply prefiltering and term breaking for documents.

3.1 Pre-Filter and Term-Breaker External Libraries

SAP IQ can use external pre-filter and term-breaker libraries written in C or C++ to prefilter and tokenize the documents during index creation or query processing. These libraries can be dynamically loaded into the process space of the database server.

> **Note**
>
> External pre-filter and term-breaker libraries must be provided by an SAP-certified partner. For information on certified vendor solutions, see the Partner Certification Reports web site and then filter the certification reports to show SAP IQ certifications.

The external dynamically loadable pre-filter and term-breaker libraries are specified in the text configuration, and need to be loaded by the database server. Each library contains an exported symbol that implements the external function specified in the text configuration object. This function returns a set of function descriptors that are used by the caller to perform the necessary tasks.

The external pre-filter and term-breaker libraries are loaded by the database server with the first `CREATE TEXT INDEX` request, when a query for a given column is received that requires the library to be loaded, or when the `TEXT` index needs to be updated.

The libraries are not loaded when an `ALTER TEXT CONFIGURATION` call is made, nor are they automatically unloaded when a `DROP TEXT CONFIGURATION` call is made. The external pre-filter and term-breaker libraries are not loaded if the server is started with the startup option to disallow the loading of external libraries.

Because these external C/C++ libraries involve the loading of non-server library code into the process space of the server, there are potential risks to data integrity, data security, and server robustness from poorly or maliciously written functions. To manage these risks, each server can explicitly enable or disable this functionality.

The `ISYSTEXTCONFIG` system view stores information about the external libraries associated with a text configuration object.

Related Information

Enable and Disable External Libraries on Startup [page 27]

3.1.1 External Library Restrictions

Text configuration objects and `TEXT` indexes using external libraries have limitations by design.

- For `TEXT` indexes on binary columns, you must use external libraries provided by external vendors for document conversion. SAP IQ does not implicitly convert documents stored in binary columns.
- N-gram based text searches are not supported if an external term-breaker is used to tokenize the document.

3.2 External Libraries on Multiplex Servers

All multiplex servers must have access to the pre-filter and term-breaker external libraries.

Users must ensure that each external pre-filter and term-breaker library is copied to the machine hosting a multiplex server and placed in a location where the server is able to load the library.

Each multiplex server works independently of other servers when calling the external pre-filter and term-breaker. Each process space can have the external libraries loaded and perform its own executions. It is assumed that the implementation of the pre-filter and term-breaker functions is the same on each server and will return the same result.

Unloading an external library from one server process space does not unload the library from other server process spaces.

3.3 Enable and Disable External Libraries on Startup

SAP IQ provides the `-sf` startup switch to enable or disable loading of external third-party libraries.

You can specify this switch in either the server startup command line or the server configuration file.

To enable the loading of external third-party libraries:

```
-sf -external_library_full_text
```

To disable the loading of external third-party libraries:

```
-sf external_library_full_text
```

To view a list of the libraries currently loaded in the server, use the `sa_list_external_library` stored procedure.

3.4 Unload External Libraries

Use the system procedure `dbo.sa_external_library_unload` to unload an external library when the library is not in use.

`dbo.sa_external_library_unload` takes one optional parameter, a `LONG VARCHAR`. The parameter specifies the name of the library to unload. If you do not specify a parameter, all external libraries that are not in use are unloaded.

Unload an external function library:

```
call sa_external_library_unload('library.dll')
```

4 Unstructured Data Queries

Learn about querying large object data, including the full text search capability that handles unstructured and semistructured data.

4.1 Full Text Search

Full text search uses `TEXT` indexes to quickly find all instances of a term (word) in a database without having to scan table rows.

`TEXT` indexes store positional information for terms in the indexed columns. Using a `TEXT` index to find rows that contain a term is faster than scanning every row in the table.

Full text search uses the `CONTAINS` search condition, which differs from searching using predicates such as `LIKE`, `REGEXP`, and `SIMILAR TO`, because the matching is term-based and not pattern-based.

String comparisons in full text search use all the normal collation settings for the database. For example, you configure the database to be case-insensitive, then full text searches are also case-insensitive.

Related Information

CONTAINS Conditions [page 47]

4.1.1 Types of Full Text Searches

Using full text search, you can search for terms, prefixes, or phrases (sequences of terms). You can also combine multiple terms, phrases, or prefixes into Boolean expressions, or use proximity searches to require that expressions appear near to each other.

Perform a full text search using a `CONTAINS` clause in either a `WHERE` clause, or a `FROM` clause of a `SELECT` statement.

4.1.1.1 Full text term and phrase searches

When performing a full text search for a list of terms, the order of terms is not important unless they are within a phrase.

If you put the terms within a phrase, the database server looks for those terms in exactly the same order, and same relative positions, in which you specified them.

When performing a term or phrase search, if terms are dropped from the query because they exceed term length settings or because they are in the stoplist, you can get back a different number of rows than you expect. This is because removing the terms from the query is equivalent to changing your search criteria. For example, if you search for the phrase `'"grown cotton"'` and grown is in the stoplist, you get every indexed row containing cotton.

You can search for the terms that are considered keywords of the CONTAINS clause grammar, as long as they are within phrases.

Term searching

In the sample database, a text index called MarketingTextIndex has been built on the Description column of the MarketingInformation table. The following statement queries the MarketingInformation.Description column and returns the rows where the value in the Description column contains the term **cotton**.

```
SELECT ID, Description
   FROM MarketingInformation
   WHERE CONTAINS ( Description, 'cotton' );
```

Table 9:

ID	Description
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** construction. Shields against sun and precipitation.cotton Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>

ID	Description
908	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>
909	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% **cotton**/20% polyester blend makes it easy to keep them clean. </body></html>
910	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying **cotton** shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist. </body></html>

The following example queries the MarketingInformation table and returns a single value for each row indicating whether the value in the Description column contains the term **cotton**.

```
SELECT ID, IF CONTAINS ( Description, 'cotton' )
      THEN 1
      ELSE 0
      ENDIF AS Results
   FROM MarketingInformation;
```

Table 10:

ID	Results
901	0
902	0
903	0

ID	Results
904	0
905	0
906	1
907	0
908	1
909	1
910	1

The next example queries the MarketingInformation table for items that have the term **cotton** the Description column, and shows the score for each match.

```
SELECT ID, ct.score, Description
   FROM MarketingInformation CONTAINS ( MarketingInformation.Description,
'cotton' ) as ct
   ORDER BY ct.score DESC;
```

Table 11:

ID	score	Description
908	0.9461597363521859	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>

ID	score	Description
910	0.9244136988525732	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Shorts</title></head><body lang=EN-US> These quick-drying **cotton** shorts provide all day comfort on or off the trails. Now with a more comfortable and stretchy fabric and an adjustable drawstring waist. </body></html>
906	0.9134171046194403	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>

ID	score	Description
909	0.8856420222728282	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Top-notch construction includes durable topstitched seams for strength with low-bulk, resilient rib-knit collar, cuffs and bottom. An 80% **cotton**/20% polyester blend makes it easy to keep them clean. </body></html>

Phrase searching

When performing a full text search for a phrase, you enclose the phrase in double quotes. A column matches if it contains the terms in the specified order and relative positions.

You cannot specify CONTAINS keywords, such as AND or FUZZY, as terms to search for unless you place them inside a phrase (single term phrases are allowed). For example, the statement below is acceptable even though NOT is a CONTAINS keyword.

```
SELECT * FROM <table-name> CONTAINS ( Remarks, '"NOT"' );
```

With the exception of asterisk, special characters are not interpreted as special characters when they are in a phrase.

Phrases cannot be used as arguments for proximity searches.

The following statement queries MarketingInformation.Description for the phrase `"grown cotton"`, and shows the score for each match:

```
SELECT ID, ct.score, Description
   FROM MarketingInformation CONTAINS ( MarketingInformation.Description,
'"grown cotton"' ) as ct
   ORDER BY ct.score DESC;
```

Table 12:

ID	score	Description
908	1.6619019465461564	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Sweatshirt</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** hooded sweatshirt with taped neck seams. Comes pre-washed for softness and to lessen shrinkage. </body></html>
906	1.6043904700786786	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** construction. Shields against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>

4.1.1.2 Full text prefix searches

The full text search feature allows you to search for the beginning portion of a term, also known as a **prefix search**.

To perform a prefix search, you specify the prefix you want to search for, followed by an asterisk. This is called a **prefix term**.

Keywords for the CONTAINS clause cannot be used for prefix searching unless they are in a phrase.

You also can specify multiple prefix terms in a query string, including within phrases (for example, `'"shi* fab"'`).

The following example queries the MarketingInformation table for items that start with the prefix shi:

```
SELECT ID, ct.score, Description
   FROM MarketingInformation CONTAINS ( MarketingInformation.Description,
'shi*' ) AS ct
   ORDER BY ct.score DESC;
```

Table 13:

ID	score	Description
906	2.295363835537917	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown cotton construction. **Shi**elds against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>
901	1.6883275743936228	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee **Shi**rt</title></head><body lang=EN-US> We've improved the design of this perennial favorite. A sleek and technical **shi**rt built for the trail, track, or sidewalk. UPF rating of 50+. </body></html>

ID	score	Description
903	1.6336529491832605	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee **Shi**rt</title></head><body lang=EN-US> A sporty, casual **shi**rt made of recycled water bottles. It will serve you equally well on trails or around town. The fabric has a wicking finish to pull perspiration away from your skin. </body></html>
902	1.6181703448678983	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee **Shi**rt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical **shi**rt is designed for high-intensity workouts in hot and humid weather. The recycled polyester fabric is gentle on the earth and soft against your skin. </body></html>

ID 906 has the highest score because the term shield occurs less frequently than shirt in the text index.

Prefix searches on GENERIC text indexes

On GENERIC text indexes, the behavior for prefix searches is as follows:

- If a prefix term is longer than the MAXIMUM TERM LENGTH, it is dropped from the query string since there can be no terms in the text index that exceed the MAXIMUM TERM LENGTH. So, on a text index with MAXIMUM TERM LENGTH 3, searching for `'red appl*'` is equivalent to searching for `'red'`.
- If a prefix term is shorter than MINIMUM TERM LENGTH, and is not part of a phrase search, the prefix search proceeds normally. So, on a GENERIC text index where MINIMUM TERM LENGTH is 5, searching

for `'macintosh a*'` returns indexed rows that contain macintosh and any terms of length 5 or greater that start with a.

- If a prefix term is shorter than MINIMUM TERM LENGTH, but is part of a phrase search, the prefix term is dropped from the query. So, on a GENERIC text index where MINIMUM TERM LENGTH is 5, searching for `'"macintosh appl* turnover"'` is equivalent to searching for macintosh followed by any term followed by turnover. A row containing `"macintosh turnover"` is not found; there must be a term between macintosh and turnover.

Prefix searches on NGRAM text indexes

On NGRAM text indexes, prefix searching can return unexpected results since an NGRAM text index contains only n-grams, and contains no information about the beginning of terms. Query terms are also broken into n-grams, and searching is performed using the n-grams not the query terms. Because of this, the following behaviors should be noted:

- If a prefix term is shorter than the n-gram length (MAXIMUM TERM LENGTH), the query returns all indexed rows that contain n-grams starting with the prefix term. For example, on a 3-gram text index, searching for `'ea*'` returns all indexed rows containing n-grams starting with ea. So, if the terms weather and fear were indexed, the rows would be considered matches since their n-grams include eat and ear, respectively.
- If a prefix term is longer than n-gram length, and is not part of a phrase, and not an argument in a proximity search, the prefix term is converted to an n-grammed phrase and the asterisk is dropped. For example, on a 3-gram text index, searching for `'purple blac*'` is equivalent to searching for `'"pur urp rpl ple" AND "bla lac"'`.
- For phrases, the following behavior also takes place:
  - If the prefix term is the only term in the phrase, it is converted to an n-grammed phrase and the asterisk is dropped. For example, on a 3-gram text index, searching for `'"purpl*"'` is equivalent to searching for `'"pur urp rpl"'`.
  - If the prefix term is in the last position of the phrase, the asterisk is dropped and the terms are converted to a phrase of n-grams. For example, on a 3-gram text index, searching for `'"purple blac*"'` is equivalent to searching for `'"pur urp rpl ple bla lac"'`.
  - If the prefix term is not in the last position of the phrase, the phrase is broken up into phrases that are ANDed together. For example, on a 3-gram text index, searching for `'"purp* blac*"'` is equivalent to searching for `'"pur urp" AND "bla lac"'`.
- If a prefix term is an argument in a proximity search, the proximity search is converted to an AND. For example, on a 3-gram text index, searching for `'red NEAR[1] appl*'` is equivalent to searching for `'red AND "app ppl"'`.

4.1.1.3 Full text proximity searches

The full text search feature allows you to search for terms that are near each other in a single column, also known as a **proximity search**.

The full text search feature allows you to search for terms that are near each other in a single column. This is called a **proximity search**. To perform a proximity search, you specify two terms with either the keywords NEAR or BEFORE between them, or the tilde (~).

You can use proximity-expression to search for terms that are near each other. For example, `<b> NEAR[2,5] <c>` searches for instances of `<b>` and `<c>` that are at most 5 and at least 2 terms away from each other. The order of terms is not significant; `<b> NEAR <c>'` is equivalent to`<c> NEAR <b>`. If NEAR is specified without distance, a default of ten terms is applied. You can specify a tilde (~) instead of `NEAR`. This is equivalent to specifying NEAR without a distance so a default of 10 terms is applied. NEAR expressions cannot be chained together (for example, `<a> NEAR[1] <b> NEAR[1] <c>`).

BEFORE is like NEAR except that the order of terms is significant. `<b> BEFORE <c>` is not equivalent to `<c> BEFORE <b>`; in the former, the term `<b>` must precede `<c>`, while in the latter, the term `<b>` must follow `<c>`. BEFORE accepts both minimum and maximum distances (like NEAR). The default minimum distance is 1. The minimum distance, if given, must be less than or equal to the maximum distance; otherwise, an error is returned.

If you do not specify a distance, the database server uses 10 as the default distance.

You can also specify a tilde (~) instead of the NEAR keyword. For example, `'<term1> ~ <term2>'`. However, you cannot specify a distance when using the tilde form; the default of ten terms is applied.

You cannot specify a phrase as an argument in proximity searches.

In a proximity search using an NGRAM text index, if you specify a prefix term as an argument, the proximity search is converted to an AND expression. For example, on a 3-gram text index, searching for `'red NEAR[1] appl*'` is equivalent to searching for `'red AND "app ppl"'`. Since this is no longer a proximity search, the search is no longer restricted to a single column in the case where multiple columns are specified in the CONTAINS clause.

Example

Suppose that you want to search MarketingInformation.Description for the term fabric within 10 terms of the term skin. You can execute the following statement.

```
SELECT ID, "contains".score, Description
   FROM MarketingInformation CONTAINS ( Description, 'fabric ~ skin' );
```

Table 14:

ID	score	Description
902	1.5572371866083279	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester **fabric** is gentle on the earth and soft against your **skin**. </body></html>

Since the default distance is 10 terms, you did not need to specify a distance. By extending the distance by one term, however, another row is returned:

```
SELECT ID, "contains".score, Description
   FROM MarketingInformation CONTAINS ( Description, 'fabric NEAR[11] skin' );
```

Table 15:

ID	score	Description
903	1.5787803210404958	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> A sporty, casual shirt made of recycled water bottles. It will serve you equally well on trails or around town. The **fabric** has a wicking finish to pull perspiration away from your **skin**. </body></html>

ID	score	Description
902	2.163125855043747	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical shirt is designed for high-intensity workouts in hot and humid weather. The recycled polyester **fabric** is gentle on the earth and soft against your **skin**. </body></html>

The score for ID 903 is higher because the terms are closer together.

4.1.1.4 Full text boolean searches

You can specify multiple terms separated by Boolean operators such as AND, OR, and AND NOT when performing full text searches.

Using the AND operator in full text searches

The AND operator matches a row if it contains both of the terms specified on either side of the AND. You can also use an ampersand (&) for the AND operator. If terms are specified without an operator between them, AND is implied.

The order in which the terms are listed is not important.

For example, each of the following statements finds rows in MarketingInformation.Description that contain the term **`fabric`** and a term that begins with **`ski`**:

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'ski* AND fabric' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric & ski*' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'ski* fabric' );
```

Using the OR operator in full text searches

The OR operator matches a row if it contains at least one of the specified search terms on either side of the OR. You can also use a vertical bar (|) for the OR operator; the two are equivalent.

The order in which the terms are listed is not important.

For example, either statement below returns rows in the MarketingInformation.Description that contain either the term **fabric** or a term that starts with **ski**:

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'ski* OR fabric' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric | ski*' );
```

Using the AND NOT operator in full text searches

The AND NOT operator finds results that match the left argument and do not match the right argument. You can also use a hyphen (-) for the AND NOT operator; the two are equivalent.

For example, the following statements are equivalent and return rows that contain the term **fabric**, but do not contain any terms that begin with **ski**.

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric AND NOT ski*' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric -ski*' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric & -ski*' );
```

Combining different boolean operators

The boolean operators can be combined in a query string. For example, the following statements are equivalent and search the MarketingInformation.Description column for items that contain **fabric** and **skin**, but not **cotton**:

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'skin fabric -cotton' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric -cotton AND
skin' );
```

The following statements are equivalent and search the MarketingInformation.Description column for items that contain **fabric** or both **cotton** and **skin**:

```
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'fabric | cotton AND
skin' );
SELECT *
   FROM MarketingInformation
   WHERE CONTAINS ( MarketingInformation.Description, 'cotton skin OR fabric' );
```

Grouping terms and phrases

Terms and expressions can be grouped with parentheses. For example, the following statement searches the MarketingInformation.Description column for items that contain **cotton** or **fabric**, and that have terms that start with **ski**.

```
SELECT ID, Description FROM MarketingInformation
   WHERE CONTAINS( MarketingInformation.Description, '( cotton OR fabric ) AND
shi*' );
```

Table 16:

ID	Description
902	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> This simple, sleek, and lightweight technical **shi**rt is designed for high-intensity workouts in hot and humid weather. The recycled polyester **fabric** is gentle on the earth and soft against your skin. </body></html>
903	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Tee Shirt</title></head><body lang=EN-US> A sporty, casual **shi**rt made of recycled water bottles. It will serve you equally well on trails or around town. The **fabric** has a wicking finish to pull perspiration away from your skin. </body></html>

ID	Description
906	<html><head><meta http-equiv=Content-Type content="text/html; charset=windows-1252"><title>Visor</title></head><body lang=EN-US> Lightweight 100% organically grown **cotton** construction. **Shi**elds against sun and precipitation. Metallic ions in the fibers inhibit bacterial growth, and help neutralize odor. </body></html>

Searching across multiple columns

You can perform a full text search across multiple columns in a single query, as long as the columns are part of the same text index.

```
SELECT *
   FROM <t>
      WHERE CONTAINS ( <t.c1>, <t.c2>, '<term1>|<term2>' );
```

```
SELECT *
   FROM <t>
   WHERE CONTAINS( <t.c1>, '<term1>' )
      OR CONTAINS( <t.c2>, '<term2>' );
```

The first query matches if `<t1.c1>` contains `<term1>`, or if `<t1.c2>` contains `<term2>`.

The second query matches if either `<t1.c1>` or `<t1.c2>` contains either `<term1>` or `<term2>`. Using the contains in this manner also returns scores for the matches.

4.1.1.5 Full text fuzzy searches

Fuzzy searching can be used to search for misspellings or variations of a word.

To do so, use the FUZZY operator followed by a string in double quotes to find an approximate match for the string. For example, `CONTAINS ( Products.Description, 'FUZZY "cotton"' )` returns `cotton` and misspellings such as `coton` or `cotten`.

> **Note**
>
> You can only perform fuzzy searches on text indexes built using the NGRAM term breaker.

Using the FUZZY operator is equivalent to breaking the string manually into substrings of length `<n>` and separating them with OR operators. For example, suppose you have a text index configured with the NGRAM term breaker and a MAXIMUM TERM LENGTH of 3. Specifying `'FUZZY "500 main street"'` is equivalent to specifying `'500 OR mai OR ain OR str OR tre OR ree OR eet'`.

The FUZZY operator is useful in a full text search that returns a score. This is because many approximate matches may be returned, but usually only the matches with the highest scores are meaningful.

4.1.1.6 Full text searches on views

To use a full text search on a view or derived table, you must build a text index on the columns in the base table that you want to perform a full text search on.

The following statements create a view on the MarketingInformation table in the sample database, which already has a text index name, and then perform a full text search on that view.

To create a view on the MarketingInformation base table, execute the following statement:

```
CREATE VIEW MarketingInfoView AS
SELECT MI.ProductID AS ProdID,
     MI."Description" AS "Desc"
FROM GROUPO.MarketingInformation AS MI
WHERE MI."ID" > 3
```

Using the following statement, you can query the view using the text index on the underlying table.

```
SELECT *
FROM MarketingInfoView
WHERE CONTAINS ( "Desc", 'Cap OR Tee*' )
```

You can also execute the following statement to query a derived table using the text index on the underlying table.

```
SELECT *
FROM (
     SELECT MI.ProductID, MI."Description"
     FROM MarketingInformation AS MI
     WHERE MI."ID" > 4 ) AS dt ( P_ID, "Desc" )
WHERE CONTAINS ( "Desc", 'Base*' )
```

> **Note**
>
> The columns on which you want to run the full text search must be included in the SELECT list of the view or derived table.

Searching a view using a text index on the underlying base table is restricted as follows:

- The view cannot contain a TOP, FIRST, DISTINCT, GROUP BY, ORDER BY, UNION, INTERSECT, EXCEPT clause, or window function.
- The view cannot contain aggregate functions.
- A CONTAINS query can refer to a base table inside a view, but not to a base table inside a view that is inside another view.

4.1.1.6.1 FROM Clause

Specifies the database tables or views involved in a SELECT statement.

Syntax

```
... FROM <table-expression> [, …]
```

Parameters

<table-expression>:{ <table-spec> | <table-expression> <join-type> <table-spec> [ON <condition>] | (<table-expression> [, ...]) }

<table-spec>:{ [<userid>.] <table-name> [[AS] <correlation-name>] | <select-statement> [AS <correlation-name> (<column-name> [, ...])] }

<contains-expression>:{<table-name> | <view-name> } CONTAINS (<column-name> [,...], <contains-query>) [[AS] <score-correlation-name>]

Usage

<contains-expression> – use the CONTAINS clause after a table name to filter the table, and return only those rows matching the full text query specified with <contains-query>.

Every matching row of the table is returned, along with a score column that can be referred to using <score-correlation-name>, if it is specified. If <score-correlation-name> is not specified, then the score column can be referred to by the default correlation name, <contains>.

With the exception of the optional correlation name argument, the CONTAINS clause takes the same arguments as the CONTAINS search condition. There must be a TEXT index on the columns listed in the CONTAINS clause.

Related Information

CONTAINS Conditions [page 47]

4.1.1.6.2 CONTAINS Conditions

Perform a full text query using the `CONTAINS` clause in the `FROM` clause of a `SELECT` statement, or by using the `CONTAINS` search condition (predicate) in a `WHERE` clause.

Both methods return the same rows; however, the `CONTAINS` clause also returns scores for the matching rows.

Syntax

```
CONTAINS ( <column-name> [,...], contains-query-string )
contains-query-string:
   simple-expression | or-expression
simple-expression:
   primary-expression | and-expression
or-expression:
   simple-expression { OR | | } contains-query-string
primary-expression:
   basic-expression
   | FUZZY " <fuzzy-expression> "
   | and-not-expression
and-expression:
   primary-expression [ AND | & ] simple-expression
and-not-expression:
   primary-expression [ AND | & ]
   { NOT | - } basic-expression
basic-expression:
   term
   | phrase
   | ( contains-query-string )
   | proximity-expression
fuzzy-expression:
   term | fuzzy-expression term
term:
   simple-term | prefix-term
prefix-term:
   simple-term*
phrase:
   " <phrase-string> "
proximity-expression:
   term ( BEFORE | NEAR )
   [ <minimum distance>, | <maximum distance> ] <term> | <term>
   {BEFORE | NEAR | ~ } <term>
phrase-string:
    term | phrase-string term
```

Parameters

simple-term a string separated by white space and special characters that represents a single indexed term (word) for which to search.

distance a positive integer.

and-expression to specify that both `<primary-expression>` and `<simple-expression>` must be found in the `TEXT` index. By default, if no operator is specified between terms or expressions, an and-

expression is assumed. For example, `'a b'` is interpreted as `'a AND b'`. An ampersand (&) can be used instead of AND, and can abut the expressions or terms on either side (for example, `'a & b'`).

and-not-expression to specify that <primary-expression> must be present in the TEXT index, but that <basic-expression> must not be found in the TEXT index. This is also known as a negation. When you use a hyphen for negation, a space must precede the hyphen, and the hyphen must abut the subsequent term. For example, `'a -b'` is equivalent to `'a AND NOT b'`; whereas for `'a - b'`, the hyphen is ignored and the string is equivalent to `'a AND b'`. `'a-b'` is equivalent to the phrase `'"a b"'`.

or-expression to specify that at least one of <simple-expression> or <contains-query-string> must be present in the TEXT index. For example, `'a|b'` is interpreted as `'a OR b'`.

fuzzy-expression to find terms that are similar to what you specify. Fuzzy matching is only supported on NGRAM TEXT indexes.

proximity-expression to search for terms that are near each other. For example, `'b NEAR[2,5] c'` searches for instances of b and c that are at most five and at least 2 terms away from each other. The order of terms is not significant; `'b NEAR c'` is equivalent to `'c NEAR b'`. If NEAR is specified without <distance>, a default of 10 terms is applied. You can specify a tilde (~) instead of NEAR. This is equivalent to specifying NEAR without a distance so a default of 10 terms is applied. NEAR expressions cannot be chained together (for example, `'a NEAR[1] b NEAR[1] c'`).

BEFORE is like NEAR, except that the order of terms is significant. `'b BEFORE c'` is not equivalent to `'c BEFORE b'`; in the former, the term `'b'` must precede `'c'` while in the latter the term `'b'` must follow `'c'`. BEFORE accepts both minimum and maximum distances like NEAR. The default minimum distance is 1. The minimum distance, if given, must be less than or equal to the maximum distance; otherwise, an error is returned.

prefix-term to search for terms that start with the specified prefix. For example, `'datab*'` searches for any term beginning with <datab>. This is also known as a prefix search. In a prefix search, matching is performed for the portion of the term to the left of the asterisk.

Usage

The CONTAINS search condition takes a column list and <contains-query-string> as arguments.

The CONTAINS search condition can be used anywhere a search condition (also referred to as predicate) can be specified, and returns TRUE or FALSE. <contains-query-string> must be a constant string, or a variable, with a value that is known at query time.

If multiple columns are specified, then they must all refer to a single base table; a TEXT index cannot span multiple base tables. You can reference the base directly in the FROM clause, or use it in a view or derived table, provided that the view or derived table does not use DISTINCT, GROUP BY, ORDER BY, UNION, INTERSECT, EXCEPT, or a row limitation.

Queries using ANSI join syntax are supported (FULL OUTER JOIN, RIGHT OUTER JOIN, LEFT OUTER JOIN), but may have suboptimal performance. Use outer joins for CONTAINS in the FROM clause only if the score column from each of the CONTAINS clauses is required. Otherwise, move CONTAINS to an ON condition or WHERE clause.

These types of queries are unsupported:

- Remote queries using a SAP SQL Anywhere table with a full TEXT index that is joined to a remote table.

- Queries using SAP IQ and SAP SQL Anywhere tables, where the full `TEXT` index to be used is on the SAP SQL Anywhere table.
- Queries using TSQL style outer join syntax (`*=*`, `=*` and `*=`).

If you use a SQL variable less than 32KB in length as a search term and the type of variable is `LONG VARCHAR`, use `CAST` to convert the variable to `VARCHAR` data type. For example:

```
SELECT * FROM tab1 WHERE CONTAINS(c1, cast(v1 AS VARCHAR(64))
```

The following warnings apply to the use of non-alphanumeric characters in query strings:

- An asterisk in the middle of a term returns an error.
- Avoid using non-alphanumeric characters (including special characters) in fuzzy-expression, because they are treated as white space and serve as term breakers.
- If possible, avoid using non-alphanumeric characters that are not special characters in your query string. Any non-alphanumeric character that is not a special character causes the term containing it to be treated as a phrase, breaking the term at the location of the character. For example, `'things we've done'` is interpreted as `'things "we ve" done'`.

Within phrases, the asterisk is the only special character that continues to be interpreted as a special character. All other special characters within phrases are treated as white space and serve as term breakers.

Interpretation of `<contains-query-string>` takes place in two main steps:

- Step 1: Interpretation of operators and precedence: During this step, keywords are interpreted as operators, and rules of precedence are applied.
- Step 2: Application of text configuration object settings: During this step, the text configuration object settings are applied to terms. Any query terms that exceed the term length settings, or that are in the stop list, are dropped.

Related Information

Fuzzy Search Over a TEXT Index [page 53]

4.1.1.6.2.1 Operator Precedence in a CONTAINS Search Condition

During query evaluation, expressions are evaluated using an order of precedence.

The order of precedence for evaluating query expressions is:

1. `FUZZY, NEAR`
2. `AND NOT`
3. `AND`
4. `OR`

4.1.1.6.2.2 Allowed Syntax for Asterisk (*)

The asterisk is used for prefix searching in a query.

An asterisk can occur at the end of the query string, or be followed by a space, ampersand, vertical bar, closing bracket, or closing quotation mark. Any other usage of asterisk returns an error.

The table "Asterisk interpretations" shows allowable asterisk usage:

Table 17: Asterisk interpretations

Query string	Equivalent to	Interpreted as
`'th*&best'`	`'th* AND best'` and `'th* best'`	Find any term beginning with th, and the term best.
`'th*|best'`	`'th* OR best'`	Find either any term beginning with th, or the term best.
`'very&(best|th*)'`	`'very AND (best OR th*)'`	Find the term very, and the term best or any term beginning with th.
`'"fast auto*"'`		Find the term fast, immediately followed by a term beginning with auto.
`'"auto* price"'`		Find a term beginning with auto, immediately followed by the term price.

> **Note**
>
> Interpretation of query strings containing asterisks varies depending on the text configuration object settings.

4.1.1.6.2.3 Allowed Syntax for Hyphen (-)

The hyphen can be used in a query for term or expression negation, and is equivalent to NOT.

Whether a hyphen is interpreted as a negation depends on its location in the query string. For example, when a hyphen immediately precedes a term or expression, it is interpreted as a negation. If the hyphen is embedded within a term, it is interpreted as a hyphen.

A hyphen used for negation must be preceded by a white space, and followed immediately by an expression.

When used in a phrase of a fuzzy expression, the hyphen is treated as white space and used as a term breaker.

The table "Hyphen interpretations" shows the allowed syntax for hyphen:

Table 18: Hyphen interpretations

Query string	Equivalent to	Interpreted as
`'the -best'`	`'the AND NOT best', 'the AND -best', 'the & -best', 'the NOT best'`	Find the term the, and not the term best.

Query string	Equivalent to	Interpreted as
`'the -(very best)'`	`'the AND NOT (very AND best)'`	Find the term the, and not the terms very and best.
`'the -"very best"'`	`'the AND NOT "very best"'`	Find the term the, and not the phrase very best.
`'alpha-numerics'`	'"alpha numerics"'	Find the term alpha, immediately followed by the term numerics.
`'wild -west'`	`'wild west'`, and `'wild AND west'`	Find the term wild, and the term west.

4.1.1.6.2.4 Allowed Syntax for Special Characters

The table "Special character interpretations" shows the allowed syntax for all special characters, except asterisk and hyphen.

The asterisk and hyphen characters are not considered special characters, if they are found in a phrase, and are dropped.

> **Note**
>
> The restrictions on specifying string literals also apply to the query string. For example, apostrophes must be within an escape sequence.

Table 19: Special character interpretations

Character or syntax	Usage examples and remarks
ampersand (&)	The ampersand is equivalent to `AND`, and can be specified as follows: • `'a & b'` • `'a &b'` • `'a& b'` • `'a&b'`
vertical bar (|)	The vertical bar is equivalent to `OR`, and can be specified as follows: • `'a| b'` • `'a |b'` • `'a | b'` • `'a|b'`

Character or syntax	Usage examples and remarks
double quotes (")	Double quotes are used to contain a sequence of terms where order and relative distance are important. For example, in the query string `'learn "full text search"'`, "full text search" is a phrase. In this example, learn can come before or after the phrase, or exist in another column (if the `TEXT` index is built on more than one column), but the exact phrase must be found in a single column.
parentheses ()	Parentheses are used to specify the order of evaluation of expressions, if different from the default order. For example, `'a AND (b|c)'` is interpreted as a, and b or c.
tilde (~)	The tilde is equivalent to `NEAR`, and has no special syntax rules. The query string `'full~text'` is equivalent to `'full NEAR text'`, and is interpreted as: the term full within ten terms of the term text.
square brackets []	Square brackets are used in conjunction with the keyword `NEAR` to contain `<distance>`. Other uses of square brackets return an error.

4.1.1.6.2.5 Effect of Dropped Terms

A `TEXT` index may exclude terms that meet certain conditions.

`TEXT` indexes are built according to the settings defined for the text configuration object used to create the `TEXT` index. A`TEXT` index excludes terms that meet any of the following conditions:

- The term is included in the stop list.
- The term is shorter than the minimum term length (`GENERIC` only).
- The term is longer than the maximum term length.

The same rules apply to query strings. The dropped term can match zero or more terms at the end or beginning of the phrase. For example, suppose the term 'the' is in the stop list:

- If the term appears on either side of an `AND`, `OR`, or `NEAR`, then both the operator and the term are removed. For example, searching for `'the AND apple'`, `'the OR apple'`, or `'the NEAR apple'` are equivalent to searching for `'apple'`.
- If the term appears on the right side of an `AND NOT`, both the `AND NOT` and the term are dropped. For example, searching for `'apple AND NOT the'` is equivalent to searching for `'apple'`.
- If the term appears on the left side of an `AND NOT`, the entire expression is dropped. For example, searching for `'the AND NOT apple'` returns no rows. Another example: `'orange and the AND NOT apple'` is the same as `'orange AND (the AND NOT apple)'` which, after the `AND NOT` expression is dropped, is equivalent to searching for `'orange'`. Contrast this with the search expression `'(orange and the) and not apple'`, which is equivalent to searching for `'orange and not apple'`.
- If the term appears in a phrase, the phrase is allowed to match with any term at the position of the dropped term. For example, searching for `'feed the dog'` matches `'feed the dog'`, `'feed my dog'`, `'feed any dog'`, and so on.

> **Note**
>
> If all of the terms for which you are searching are dropped, SAP IQ returns the error `CONTAINS has NULL search term`. SAP SQL Anywhere reports no error and returns zero rows.

4.1.1.6.2.6 Query Match Score

You can sort query results using the score that indicates the closeness of a match.

When you include a `CONTAINS` clause in the `FROM` clause of a query, each match has a score associated with it. The score indicates how close the match is, and you can use score information to sort the data. Two main criteria determine score:

- The number of times a term appears in the indexed row. The more times a term appears in an indexed row, the higher its score.
- The number of times a term appears in the `TEXT` index. The more times a term appears in a `TEXT` index, the lower its score.

Depending on the type of full text search, other criteria affect scoring. For example, in proximity searches, the proximity of search terms impacts scoring. By default, the result set of a `CONTAINS` clause has the correlation name `contains` that has a single column in it called `score`. You can refer to `"contains".score` in the `SELECT` list, `ORDER BY` clause, or other parts of the query. However, because `contains` is a SQL reserved word, you must remember to put it in double quotes. Alternatively, you can specify another correlation name, for example, `CONTAINS ( expression ) AS ct`. The examples for full text search refer to the `score` column as `ct.score`.

This statement searches `MarketingInformation.Description` for terms starting with ‘stretch” or terms starting with “comfort”:

```
SELECT ID, ct.score, Description
FROM MarketingInformation
CONTAINS ( MarketingInformation.Description,
           'stretch* | comfort*' )
AS ct ORDER BY ct.score DESC;
```

4.2 NGRAM TEXT Index Searches

Fuzzy and non-fuzzy search capability over a `TEXT` index is possible for `TEXT` indexes of type `NGRAM`.

4.2.1 Fuzzy Search Over a TEXT Index

Fuzzy search capability over a `TEXT` index is possible only if the `TEXT` index is of type `NGRAM`. The `GENERIC TEXT` index cannot handle the fuzzy search.

Fuzzy searching can be used to search for misspellings or variations of a word. To do so, use the `FUZZY` operator followed by a string in double quotes to find an approximate match for the string.

Using the `FUZZY` operator is equivalent to breaking the string manually into substrings of length `<n>` and separating them with `OR` operators. For example, if you have a text index configured with the `NGRAM` term breaker and a `MAXIMUM TERM LENGTH` of 3, specifying `'FUZZY "500 main street" '` is equivalent to specifying `'500 OR mai OR ain OR str OR tre OR ree OR eet'`.

The FUZZY operator is useful in a full text search that returns a score. Many approximate matches may be returned, but usually only the matches with the highest scores are meaningful.

Note

Fuzzy search does not support prefix or suffix searching. For example, the search clause cannot be “v*” or “*vis”.

Example

Fuzzy search over an NGRAM TEXT index

Create a table and an NGRAM TEXT index:

```
CREATE TEXT CONFIGURATION NGRAMTxtcfg
   FROM default_char;
ALTER TEXT CONFIGURATION NGRAMTxtcfg TERM BREAKER    NGRAM;
ALTER TEXT CONFIGURATION NGRAMTxtcfg maximum term    length 3;
CREATE TABLE t_iq(a int, b varchar(100));
CREATE TEXT INDEX TXT_IQ on t_iq(b) CONFIGURATION    NGRAMTxtcfg
```

Insert this data into the table:

```
INSERT INTO t_iq values (1,'hello this is hira ');
INSERT INTO t_iq values(2, ' book he ookw worm okwo
kwor');
INSERT INTO t_iq values(3,'Michael is a good person');
INSERT INTO t_iq values(4,'hello this is evaa');
INSERT INTO t_iq values(5,'he is a bookworm');
INSERT INTO t_iq values (6,'boo ook okw kwo wor orm');
```

After inserting the data, execute this query to perform fuzzy searching over an NGRAM TEXT index:

```
SELECT * FROM t_iq WHERE CONTAINS (b,'FUZZY "bookerm"');
```

The results of the query are:

```
a         b
2         book he ookw worm okwo kwor
5         he is a bookworm
6         boo ook okw kwo wor orm
```

Example

Additional letter in the fuzzy search clause

This query illustrates an additional letter in the fuzzy search clause:

```
SELECT * FROM t_iq WHERE CONTAINS (b,'FUZZY "hellow"');
```

The results of the query are:

```
a         b
1         hello this is hira
4         hello this is evaa
```

Example

Letter removed from the fuzzy search clause

In this query, a letter is removed from the fuzzy search clause:

```
SELECT * FROM t_iq WHERE CONTAINS(b, 'FUZZY "hllo"');
```

The results of the query are:

```
a          b
1          hello this is hira
4          hello this is evaa
```

4.2.2 Non-fuzzy Search Over a TEXT Index

Non-fuzzy search on `NGRAM` breaks the term into corresponding n-grams and searches for the n-grams in the `NGRAM TEXT` index.

The query `CONTAINS ( M.Description, 'ams' ) ct;` illustrates a non-fuzzy `NGRAM` search over a 2GRAM index, which is semantically equal to searching query `CONTAINS( M.Description, '"am ms"' ) ct;`

If you search for a 'v*' TERM on a 2GRAM index, then v followed by any alphabet is considered as a matching 2GRAM for the searching term and is output as a result.

The query `CONTAINS (M.Description, 'white whale') ct;` illustrates a non-fuzzy `NGRAM` search over a 3GRAM index and is semantically equal to searching query `CONTAINS (M.Description, '"whi hit ite wha hal ale"');`

The difference between `NGRAM` fuzzy and non-fuzzy search is that fuzzy search is a disjunction over individual GRAMS. Non-fuzzy search is a conjunction over the individual GRAMS. When `GENERIC` and `NGRAM TEXT` indexes are created on the same column, then the `GENERIC TEXT` index is used for a query with non-fuzzy search and the `NGRAM TEXT` index is used for fuzzy search.

Example

Non-fuzzy search after creating a `GENERIC TEXT` index on the same column

This query illustrates non-fuzzy search after creating a `GENERIC TEXT` index on the same column:

```
SELECT * FROM t_iq WHERE CONTAINS (b,'bookworm');
```

The results of the query are:

```
a          b
5          he is a bookworm
```

Example

Fuzzy search with both `NGRAM` and `GENERIC TEXT` indexes on the same column

This query illustrates fuzzy search with both `NGRAM` and `GENERIC TEXT` indexes on the same column:

```
SELECT * FROM t_iq
WHERE CONTAINS (b,'FUZZY "bookwerm"');
```

The results of the query are:

```
a         b
2         book he ookw worm okwo kwor
5         he is a bookworm
6         boo ook okw kwo wor orm
```

Example

Fuzzy search phrase in a non-fuzzy search clause

This query illustrates the behavior of a fuzzy search phrase in a non-fuzzy search clause:

```
SELECT * FROM t_iq WHERE CONTAINS (b,'bookwerm');
```

No result is returned for this query.

4.3 Queries on LONG BINARY Columns

In `WHERE` clauses of the `SELECT` statement, you can use `LONG BINARY` columns only in IS NULL and IS NOT NULL expressions, in addition to the `BYTE_LENGTH64`, `BYTE_SUBSTR64`, `BYTE_SUBSTR`, `BIT_LENGTH`, `OCTET_LENGTH`, `CHARINDEX`, and `LOCATE` functions.

You cannot use `LONG BINARY` columns in the `SELECT` statement clauses `ORDER BY`, `GROUP BY`, and `HAVING` or with the `DISTINCT` keyword.

SAP IQ does not support `LIKE` predicates on `LONG BINARY` (`BLOB`) columns or variables. Searching for a pattern in a `LONG BINARY` column using a `LIKE` predicate returns the error `Invalid data type comparison in predicate`.

Related Information

Function Support [page 93]

4.4 Queries on LONG VARCHAR Columns

In `WHERE` clauses of the `SELECT` statement, you can use `LONG VARCHAR` columns only in IS NULL and IS NOT NULL expressions, in addition to the `BIT_LENGTH`, `CHAR_LENGTH`, `CHAR_LENGTH64`, `CHARINDEX`, `LOCATE`, `OCTET_LENGTH`, `PATINDEX`, `SUBSTRING64`, and `SUBSTRING` functions.

You can use the `LIKE` predicate to search for a pattern on a `LONG VARCHAR` column. All patterns of 126 or fewer characters are supported. Patterns longer than 254 characters are not supported. Some patterns between 127 and 254 characters in length are supported, depending on the contents of the pattern.

The `LIKE` predicate supports `LONG VARCHAR` (`CLOB`) variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.

You cannot use `LONG VARCHAR` columns in the `SELECT` statement clauses `ORDER BY`, `GROUP BY`, and `HAVING` or with the `DISTINCT` keyword (`SELECT DISTINCT` and `COUNT DISTINCT`).

Related Information

Function Support [page 93]

4.4.1 CONTAINS Predicate Support

You can create a `WORD` (`WD`) index on a `LONG VARCHAR` (`CLOB`) column and use the `CONTAINS` predicate to search the column for string constants of maximum length 255 characters.

The `CONTAINS` predicate is not supported on `LONG BINARY` (`BLOB`) columns using `WD` indexes. If you attempt to search for a string in a `LONG BINARY` column with a `WD` index using a `CONTAINS` predicate, an error is returned. `TEXT` indexes that use an external library support `CONTAINS` on binary data.

4.5 Performance Monitoring of LONG BINARY and LONG VARCHAR Columns

The performance monitor displays performance data for `LONG BINARY` and `LONG VARCHAR` columns.

5 Stored Procedure Support

Learn about stored procedure support for the LONG BINARY (BLOB) and LONG VARCHAR (CLOB) data type columns and full text searching.

5.1 Term Management in a TEXT Index

You can use stored procedures to break strings into terms, to discover how many terms are in the TEXT index and their position, and to display statistical information about the TEXT indexes.

5.1.1 sa_char_terms System Procedure

Breaks a CHAR string into terms and returns each term as a row along with its position.

Syntax

```
sa_char_terms( '<char-string>' [, <'text-config-name>' [, '<owner>' ] ] ]     )
```

Parameters

char-string the CHAR string you are parsing.

text-config-name the text configuration object to apply when processing the string. The default value is 'default_char'.

owner the owner of the specified text configuration object. The default value is DBA.

Privileges

None

Remarks

You can use `sa_char_terms` to find out how a string is interpreted when the settings for a text configuration object are applied. This can be helpful when you want to know what terms would be dropped during indexing or from a query string.

Example

Return the terms in the `CHAR` string 'the quick brown fox jumped over the fence':

```
CALL sa_char_terms
( 'the quick brown fox jumped over the fence' );
```

Table 20: CHAR string interpretation

Term	Position
the	1
quick	2
brown	3
fox	4
jumped	5
over	6
the	7
fence	8

5.1.2 sa_nchar_terms System Procedure

Breaks an `NCHAR` string into terms and returns each term as a row along with its position.

Syntax

```
sa_nchar_terms( '<char-string>' [ , <'text-config-name>' [, '<owner>' ] ] ]     )
```

Parameters

char-string the `NCHAR` string you are parsing.
text-config-name the text configuration object to apply when processing the string. The default value is 'default_nchar'.
owner the owner of the specified text configuration object. The default value is DBA.

Privileges

You must have EXECUTE privilege on the system procedure.

Remarks

You can use `sa_nchar_terms` to find out how a string is interpreted when the settings for a text configuration object are applied. This can be helpful when you want to know what terms would be dropped during indexing or from a query string.

The syntax for `sa_nchar_terms` is similar to the syntax for the `sa_char_terms` system procedure.

> **Note**
>
> The `NCHAR` data type is supported only for `IN SYSTEM` tables.

5.1.3 sa_text_index_stats System Procedure

Returns statistical information about the TEXT indexes in the database.

Syntax

```
sa_text_index_stats( )
```

Privileges

You must have EXECUTE privilege on the system procedure. You must also have one of the following system privileges:

- MANAGE ANY STATISTICS
- CREATE ANY INDEX
- ALTER ANY INDEX
- DROP ANY INDEX
- CREATE ANY OBJECT
- ALTER ANY OBJECT
- DROP ANY OBJECT

Remarks

Use `sa_text_index_stats` to view statistical information for each TEXT index in the database.

Table 21: Statistical information for TEXT indexes returned by sa_text_index_stats

Column name	Type	Description
owner_id	UNSIGNED INT	ID of the owner of the table
table_id	UNSIGNED INT	ID of the table
index_id	UNSIGNED INT	ID of the TEXT index
text_config_id	UNSIGNED BIGINT	ID of the text configuration referenced by the TEXT index
owner_name	CHAR(128)	Name of the owner
table_name	CHAR(128)	Name of the table
index_name	CHAR(128)	Name of the TEXT index
text_con-fig_name	CHAR(128)	Name of the text configuration object
doc_count	UNSIGNED BIGINT	Total number of indexed column values in the TEXT index
doc_length	UNSIGNED BIGINT	Total length of data in the TEXT index
pending_length	UNSIGNED BIGINT	Total length of the pending changes
deleted_length	UNSIGNED BIGINT	Total length of the pending deletions
last_refresh	TIMESTAMP	Date and time of the last refresh

The `pending_length`, `deleted_length`, and `last_refresh` values are NULL for IMMEDIATE REFRESH TEXT indexes.

Example

Return statistical information for each TEXT index in the database:

```
CALL sa_text_index_stats( );
```

5.1.4 sa_text_index_vocab System Procedure

Lists all terms that appear in a TEXT index, and the total number of indexed values in which each term appears.

Syntax

```
sa_text_index_vocab(
    '<text-index-name>',
    '<table-name>',
    '<table-owner>'
    )
```

Parameters

text-index-name use this `CHAR`(128) parameter to specify the name of the TEXT index.
table-name use this `CHAR`(128) parameter to specify the name of the table on which the TEXT index is built.
table-owner use this `CHAR`(128) parameter to specify the owner of the table.

Privileges

You must have EXECUTE privilege on the system procedure. You must also have one of the following:

- SELECT ANY TABLE system privilege
- SELECT privilege on the indexed table

Remarks

`sa_text_index_vocab` returns all terms that appear in a TEXT index, and the total number of indexed values in which each term appears (which is less than the total number of occurrences, if the term appears multiple times in some indexed values).

Parameter values cannot be host variables or expressions. The arguments `<text-index-name>`, `<table-name>`, and `<table-owner>` must be constraints or variables.

Example

Execute `sa_text_index_vocab` to return all the terms that appear in the TEXT index `MyTextIndex` on table `Customers` owned by `GROUPO`:

```
sa_text_index_vocab
('MyTextIndex','Customers','GROUPO');
```

Table 22: Terms in the index

term	freq
a	1
Able	1
Acres	1
Active	5
Advertising	1
Again	1
...	...

5.2 External Library Identification

The `sa_list_external_library` stored procedure lists the libraries that are currently loaded in the server. Once identified, use `sa_external_library_unload` to unload the library from the server.

5.2.1 sa_external_library_unload System Procedure

Unloads an external library.

Syntax

```
sa_external_library_unload ( [ '<external-library'> ] )
```

Parameters

external-library optionally use this `LONG VARCHAR` parameter to specify the name of a library to be unloaded. If no library is specified, all external libraries that are not in use are unloaded.

Privileges

You must have EXECUTE privilege on the system procedure. You must also have the MANAGE ANY EXTERNAL OBJECT system privilege.

Remarks

If an external library is specified, but is in use or is not loaded, an error is returned. If no parameter is specified, an error is returned if no loaded external libraries are found.

Example

Unload an external library called `myextlib.dll`:

```
CALL sa_external_library_unload( 'myextlib.dll' );
```

Unload all libraries that are not currently in use:

```
CALL sa_external_library_unload();
```

5.2.2 sa_list_external_library System Procedure

Lists the external libraries currently loaded in the server.

Syntax

```
sa_list_external_library( )
```

Privileges

You must have EXECUTE privilege on the system procedure. You must also have the MANAGE ANY EXTERNAL OBJECT system privilege.

Remarks

Returns a list of external libraries loaded in the engine along with their reference count.

The reference count is the number of instances of the library in the engine. An external library can be unloaded by executing the procedure `sa_external_library_unload`, only if its reference count is 0.

Example

List the external libraries and their reference count:

```
CALL sa_list_external_library()
```

5.3 Large Object Data Compression

The `sp_iqsetcompression` stored procedure controls the compression of large object columns.

`sp_iqsetcompression` controls the compression of columns of data type `LONG BINARY` and `LONG VARCHAR` when writing database buffers to disk. You can also use `sp_iqsetcompression` to disable compression. This functionality saves CPU cycles, because certain data formats stored in a `LONG BINARY` or `LONG VARCHAR` column (for example, JPG files) are already compressed and gain nothing from additional compression.

The `sp_iqshowcompression` stored procedure displays the compression setting of large object columns.

5.3.1 sp_iqsetcompression Procedure

Sets compression of data in columns of `LONG BINARY` (`BLOB`) and `LONG VARCHAR` (`CLOB`) data types.

Syntax

```
sp_iqsetcompression ( <owner>, <table>, <column>, <on_off_flag> )
```

Permissions

You must have EXECUTE privilege on the system procedure. You must also have one of the following system privilege:

- ALTER ANY TABLE
- ALTER ANY OBJECT

Remarks

`sp_iqsetcompression` provides control of compression of `LONG BINARY` (`BLOB`) and `LONG VARCHAR` (`CLOB`) data type columns. The compression setting applies only to base tables.

A side effect of `sp_iqsetcompression` is that a `COMMIT` occurs after you change the compression setting.

Table 23: sp_iqsetcompression parameters

Name	Description
`<owner>`	Owner of the table for which you are setting compression
`<table>`	Table for which you are setting compression
`<column>`	Column for which you are setting compression
`<on_off_flag>`	Compression setting: ON enables compression, OFF disables compression

Example

Assume this table definition:

```
CREATE TABLE USR.pixTable (picID INT NOT NULL,
picJPG LONG BINARY NOT NULL);
```

To turn off compression on the LOB column `picJPG`, call `sp_iqsetcompression`:

```
CALL sp_iqsetcompression('USR', 'pixTable', 'picJPG',
'OFF') ;
```

This command returns no rows.

5.3.2 sp_iqshowcompression Procedure

Displays compression settings for columns of `LONG BINARY` (`BLOB`) and `LONG VARCHAR` (`CLOB`) data types.

Syntax

```
sp_iqshowcompression ( <owner>, <table>, <column> )
```

Privileges

You must have EXECUTE privilege on the system procedure. You must also have one of the following system privileges:

- ALTER ANY TABLE
- ALTER ANY OBJECT

Remarks

Returns the column name and compression setting. Compression setting values are 'ON' (compression enabled) and 'OFF' (compression disabled).

Table 24: sp_iqshowcompression parameters

Name	Description
`<owner>`	Owner of the table for which you are setting compression
`<table>`	Table for which you are setting compression
`<column>`	Column for which you are setting compression

Example

Assume this table definition:

```
CREATE TABLE USR.pixTable (picID INT NOT NULL,
picJPG LONG BINARY NOT NULL);
```

To check the compression status of the columns in the `pixTable` table, call `sp_iqshowcompression`:

```
CALL sp_iqshowcompression('USR', 'pixTable',
```

```
'picJPG') ;
```

This command returns one row:

```
'picJPG','ON'
```

5.4 Information About Large Object Columns

The stored procedure `sp_iqindexsize` displays the size of an individual `LONG BINARY` and `LONG VARCHAR` column.

5.4.1 Size of a LONG BINARY Column

`sp_iqindexsize` output that shows a `LONG BINARY` column with approximately 42GB of data.

The page size is 128KB. The `largelob` Info type is in the last row.

```
Username    Indexname              Type Info     KBytes   Pages   Compressed
                                                                    Pages

DBA    test10.DBA.ASIQ_IDX_T128_C3_FP FP Total     42953952  623009  622923
DBA    test10.DBA.ASIQ_IDX_T128_C3_FP FP vdo       0         0       0
DBA    test10.DBA.ASIQ_IDX_T128_C3_FP FP bt        0         0       0
DBA    test10.DBA.ASIQ_IDX_T128_C3_FP FP garray    0         0       0
DBA    test10.DBA.ASIQ_IDX_T128_C3_FP FP bm        136       2       1
DBA    test10.DBA.ASIQ_IDX_T128_C3_FP FP barray    2312      41      40
DBA    test10.DBA.ASIQ_IDX_T128_C3_FP FP dpstore   170872    2551    2549
DBA    test10.DBA.ASIQ_IDX_T128_C3_FP FP largelob  42780632  620415  620333
```

In this example, the compression ratio is 42953952/(623009*128) = 53.9%.

5.4.2 Size of a LONG VARCHAR Column

`sp_iqindexsize` output that shows a `LONG VARCHAR` column with approximately 42GB of data.

The page size is 128KB. The `largelob` Info type is in the last row.

```
Username    Indexname              Type Info     KBytes   Pages   Compressed
                                                                    Pages
```

DBA	test10.DBA.ASIQ_IDX_T128_C3_FP FP Total	42953952	623009	622923
DBA	test10.DBA.ASIQ_IDX_T128_C3_FP FP vdo	0	0	0
DBA	test10.DBA.ASIQ_IDX_T128_C3_FP FP bt	0	0	0
DBA	test10.DBA.ASIQ_IDX_T128_C3_FP FP garray	0	0	0
DBA	test10.DBA.ASIQ_IDX_T128_C3_FP FP bm	136	2	1
DBA	test10.DBA.ASIQ_IDX_T128_C3_FP FP barray	2312	41	40
DBA	test10.DBA.ASIQ_IDX_T128_C3_FP FP dpstore	170872	2551	2549
DBA	test10.DBA.ASIQ_IDX_T128_C3_FP FP largelob	42780632	620415	620333

In this example, the compression ratio is 42953952/(623009*128) = 53.9%.

6 Large Object Data Load and Unload

Learn how to export and load large object data.

6.1 Large Object Data Exports

The SAP IQ data extraction facility includes the `BFILE` function, which allows you to extract individual `LONG BINARY` and `LONG VARCHAR` cells to individual operating system files on the server.

You can use `BFILE` with or without the data extraction facility.

6.1.1 BFILE Function

Extracts individual `LONG BINARY` and `LONG VARCHAR` cells to individual operating system files on the server.

Syntax

```
BFILE( <file-name-expression>, <large-object-column> )
```

Parameters

file-name-expression the name of the output file into which the `LONG BINARY` or `LONG VARCHAR` data is written. This file name can be up to (32K -1) bytes in length, but must be a valid path name that is supported by the file system.
large-object-column the name of the `LONG BINARY` or `LONG VARCHAR` column.

Usage

`BFILE` returns:

- 1, if the file is successfully written
- 0, if the file is not successfully opened or written

- NULL, if the LONG BINARY or LONG VARCHAR cell value is NULL

If the LONG BINARY or LONG VARCHAR cell value is NULL, no file is opened and no data is written.

The file path is relative to where the server was started and the open and write operations execute with the permissions of the server process. Tape devices are not supported for the BFILE output file.

LONG BINARY and LONG VARCHAR cells retrieved other than with the BFILE function (that is, retrieved through the client/server database connection later) are limited in size to a maximum length of 2GB. Use SUBSTRING64 or BYTE_SUBSTR64 to retrieve LONG BINARY cells greater than 2GB using a SELECT (SELECT, OPEN CURSOR). Use SUBSTRING64 to retrieve LONG VARCHAR cells greater than 2GB using a SELECT (SELECT, OPEN CURSOR). Some connection drivers, for example ODBC, JDBC™, and Open Client™, do not allow more than 2GB to be returned in one SELECT.

You can use BFILE with or without the data extraction facility.

6.1.1.1 BFILE Function Example

Use BFILE to extract and reload LOB data.

Create table LobA:

```
create table LobA
  (rowid  int primary key,
    col1   clob null,
    col2   blob null)
```

Assume LobA has two rows of data.

Extract the non-LOB data and the paths to the files into which the LOB data is extracted:

```
BEGIN
  SET TEMPORARY OPTION
    Temp_Extract_Name1 = LobA_data.txt';
  SELECT rowid,
    'row' + string(rowid) + '.' + 'col1',
    'row' + string(rowid) + '.' + 'col2'
  FROM LobA;
END
```

The file LobA_data.txt is created and contains this non-LOB data and these filenames:

```
  1,row1.col1,row1.col2,
  2,row2.col1,row2.col2,
```

Perform the LOB data extraction:

```
SELECT
  BFILE('row' + string(rowid) + '.' + 'col1',col1),
  BFILE('row' + string(rowid) + '.' + 'col2',col2)
FROM LobA;
```

After the extraction, there is a file for each cell of LOB data extracted. For example, if table LobA contains two rows of data with rowid values of 1 and 2, you have these files:

- row1.col1
- row1.col2
- row2.col1
- row2.col2

Reload the extracted data:

```
LOAD TABLE LobA
(rowid,
  col1 ASCII FILE (',') NULL('NULL'),
  col2 BINARY FILE (',') NULL('NULL'))
FROM LobA_data.txt'
DELIMITED BY ','
ROW DELIMITED BY '\n'
ESCAPES OFF;
```

6.2 Large Object Data Loads

Load `LONG BINARY` and `LONG VARCHAR` data using the extended syntax of the `LOAD TABLE` statement.

You can load large object data of unlimited size, unless restricted by the operating system, from a primary file in ASCII or BCP format. The maximum length of fixed-width data loaded from a primary file into large object columns is 32K - 1.

You can also specify a secondary load file in the primary load file. Each individual secondary data file contains exactly one `LONG BINARY` or `LONG VARCHAR` cell value.

6.2.1 Extended LOAD TABLE Syntax

`LOAD TABLE` has extended syntax for loading large object data.

```
LOAD [ INTO ] TABLE [ <owner> ].<table-name>
... ( <column-name> <load-column-specification> [, ...] )
... FROM '<filename>-<string>' [, ...]
... [ QUOTES { ON | OFF } ]
... ESCAPES OFF
... [ FORMAT { ascii | binary | bcp } ]
... [ DELIMITED BY '<string>' ]
...
```

```
<load-column-specification>:
...
| { BINARY | ASCII } FILE( <integer> )
| { BINARY | ASCII } FILE ( '<string>' )
```

The keywords `BINARY FILE` (for `LONG BINARY`) or `ASCII FILE` (for `LONG VARCHAR`) specify to the load that the primary input file for the column contains the path of the secondary file (which contains the `LONG BINARY` or `LONG VARCHAR` cell value), rather than the `LONG BINARY` or `LONG VARCHAR` data itself. The secondary file pathname can be either fully qualified or relative. If the secondary file pathname is not fully qualified, then the

path is relative to the directory in which the server was started. Tape devices are not supported for the secondary file.

SAP IQ supports loading `LONG BINARY` and `LONG VARCHAR` values of unlimited length (subject to operating system restrictions) in the primary load file. When binary data of hexadecimal format is loaded into a `LONG BINARY` column from a primary file, SAP IQ requires that the total number of hexadecimal digits is an even number. The error "`Odd length of binary data value detected on column`" is reported, if the cell value contains an odd number of hexadecimal digits. Input files for `LONG BINARY` loads should always contain an even number of hexadecimal digits.

SAP IQ does not support loading large object columns from primary files using `LOAD TABLE…FORMAT BINARY`. You can load large object data in binary format from secondary files.

For `LOAD TABLE FORMAT BCP`, the load specification may contain only column names, `NULL`, and `ENCRYPTED`. This means that you cannot use secondary files when loading `LONG BINARY` and `LONG VARCHAR` columns using the `LOAD TABLE FORMAT BCP` option.

6.2.2 Large Object Data Load Example

Create and load a table with `LONG BINARY` data.

```
CREATE TABLE ltab (c1 INT, filename CHAR(64),
    ext CHAR(6), lobcol LONG BINARY NULL);
```

```
LOAD TABLE ltab (
    c1,
    filename,
    ext NULL('NULL'),
    lobcol BINARY FILE (',') NULL('NULL')
)
FROM 'abc.inp'
QUOTES OFF ESCAPES OFF;
```

The primary file `abc.inp` contains this data:

```
1,boston,jpg,/s1/loads/lobs/boston.jpg,
2,map_of_concord,bmp,/s1/loads/maprs/concord.bmp,
3,zero length test,NULL,,
4,null test,NULL,NULL,
```

After the `LONG BINARY` data is loaded into table `tab`, the first and second rows for column `lobcol` contain the contents of files `boston.jpg` and `concord.bmp`, respectively. The third and fourth rows contain a zero-length value and NULL, respectively.

6.2.3 Control of Load Errors

The database option SECONDARY_FILE_ERROR allows you to specify the action of the load if an error occurs while opening or reading from a secondary BINARY FILE or ASCII FILE.

If SECONDARY_FILE_ERROR is ON, the load rolls back if an error occurs while opening or reading from a secondary BINARY FILE or ASCII FILE.

If SECONDARY_FILE_ERROR is OFF (the default), the load continues, regardless of any errors that occur while opening or reading from a secondary BINARY FILE or ASCII FILE. The LONG BINARY or LONG VARCHAR cell is left with one of these values:

- NULL, if the column allows nulls
- Zero-length value, if the column does not allow nulls

Any user can set SECONDARY_FILE_ERROR for the PUBLIC role or temporary; the option setting takes effect immediately.

When logging integrity constraint violations to the load error ROW LOG file, the information logged for a LONG BINARY or LONG VARCHAR column is:

- Actual text as read from the primary data file, if the logging occurs within the first pass of the load operation
- Zero-length value, if the logging occurs within the second pass of the load operation

6.2.4 Load of Large Object Data with Trailing Blanks

The LOAD TABLE...STRIP option has no effect on LONG VARCHAR data.

Trailing blanks are not stripped from LONG VARCHAR data, even if the STRIP option is on.

6.2.5 Load of Large Object Data with Quotes

The LOAD TABLE...QUOTES option does not apply to loading LONG BINARY (BLOB) or LONG VARCHAR (CLOB) data from the secondary file, regardless of its setting,

A leading or trailing quote is loaded as part of CLOB data. Two consecutive quotes between enclosing quotes are loaded as two consecutive quotes with the QUOTES ON option.

6.2.6 Truncation of Partial Multibyte Character Data

Partial multibyte LONG VARCHAR data is truncated during the load according to the value of the TRIM_PARTIAL_MBC database option.

- If TRIM_PARTIAL_MBC is ON, a partial multibyte character is truncated for both primary data and the LOAD with ASCII FILE option.

- If `TRIM_PARTIAL_MBC` is OFF, the `LOAD` with `ASCII FILE` option handles the partial multibyte character according to the value of the `SECONDARY_FILE_ERROR` database option.

The table "Partial multibyte character on loading LONG VARCHAR with ASCII FILE option" lists how a trailing multibyte character is loaded, depending on the values of `TRIM_PARTIAL_MBC` and `SECONDARY_FILE_ERROR`.

Table 25: Partial multibyte character on loading LONG VARCHAR with ASCII FILE option

TRIM_PARTIAL_MBC	SECONDARY_FILE_ERROR	Trailing Partial Multibyte Character Found
ON	ON/OFF	Trailing partial multibyte character truncated
OFF	ON	Cell — null, if null allowed LOAD error — roll back, if null not allowed
OFF	OFF	Cell — null, if null allowed Cell — zero-length, if null not allowed

6.2.7 Load Support of Large Object Variables

Large object variables are supported by the `LOAD TABLE`, `INSERT…VALUES`, `INSERT…SELECT`, `INSERT…LOCATION`, `SELECT…INTO`, and `UPDATE` SQL statements.

Related Information

Large Object Data Types [page 76]
Large Object Variables [page 78]

7 Large Object Data Types

Learn about the characteristics of the large object LONG BINARY and LONG VARCHAR data type columns and index support of large object data.

7.1 Large Object Data Types LONG BINARY and BLOB

Binary large object (BLOB) data in SAP IQ is stored in columns of data type LONG BINARY or BLOB.

An individual LONG BINARY data value can have a length ranging from zero (0) to 512TB (terabytes) for an IQ page size of 128KB, or 2PB (petabytes) for an IQ page size of 512KB. (The maximum length is equal to 4GB multiplied by the database page size.) To accommodate a table with LONG BINARY data, an IQ database must be created with an IQ page size of at least 128KB (131072 bytes).

A table or database can contain any number of LONG BINARY columns up to the supported maximum columns per table and maximum columns per database, respectively.

LONG BINARY columns can be either NULL or NOT NULL and can store zero-length values. The domain BLOB is a LONG BINARY data type that allows NULL.

You cannot construct a non-FP index on a LONG BINARY column.

Prefetching is disabled, if the result set contains BLOB columns.

Modify LONG BINARY columns using the UPDATE, INSERT, LOAD TABLE, DELETE, TRUNCATE, SELECT...INTO and INSERT...LOCATION SQL statements. Positioned updates and deletes are not supported on LONG BINARY columns.

You can insert an SAP ASE IMAGE column into a LONG BINARY column using the INSERT...LOCATION command. All IMAGE data inserted is silently right-truncated at 2147483648 bytes (2GB).

7.1.1 LONG BINARY Data Type Conversion

There are limited implicit data type conversions to and from the LONG BINARY data type and non-LONG BINARY data types.

There are no implicit data type conversions from the LONG BINARY data type to another non-LONG BINARY data type, except to the BINARY and VARBINARY data types for INSERT and UPDATE. There are implicit conversions to LONG BINARY data type from TINYINT, SMALLINT, INTEGER, UNSIGNED INTEGER, BIGINT, UNSIGNED BIGINT, CHAR, and VARCHAR data types. There are no implicit conversions from BIT, REAL, DOUBLE, or NUMERIC data types to LONG BINARY data type. Implicit conversion can be controlled using the CONVERSION_MODE database option.

The currently supported byte substring functions for the LONG BINARY data type are accepted as input for implicit conversion for the INSERT and UPDATE statements.

The LONG BINARY data type can be explicitly converted to BINARY or VARBINARY. No other explicit data type conversions (for example, using the CAST or CONVERT function) exist either to or from the LONG BINARY data type.

Truncation of LONG BINARY data during conversion of LONG BINARY to BINARY or VARBINARY is handled the same way the truncation of BINARY and VARBINARY data is handled. If the STRING_RTRUNCATION option is ON, any right-truncation (of any values, not just non-space characters) on INSERT or UPDATE of a binary column results in a truncation error and the transaction is rolled back.

Related Information

Function Support [page 93]

7.2 Large Object Data Types LONG VARCHAR and CLOB

Character large object (CLOB) data in SAP IQ is stored in columns of data type LONG VARCHAR or CLOB.

An individual LONG VARCHAR data value can have a length ranging from zero (0) to 512TB (terabytes) for an IQ page size of 128KB, or 2PB (petabytes) for an IQ page size of 512KB. (The maximum length is equal to 4GB multiplied by the database page size.) To accommodate a table with LONG VARCHAR data, an IQ database must be created with an IQ page size of at least 64KB (65536 bytes).

A table or database can contain any number of LONG VARCHAR columns up to the supported maximum columns per table and maximum columns per database, respectively.

SAP IQ supports both single-byte and multibyte LONG VARCHAR data.

LONG VARCHAR columns can be either NULL or NOT NULL, and can store zero-length values. The domain CLOB is a LONG VARCHAR data type that allows NULL. To create a non-null LONG VARCHAR column, explicitly specify NOT NULL in the column definition.

You can create a LONG VARCHAR column using the domain CLOB, when you create a table or add a column to an existing table. For example:

```
CREATE TABLE lvtab (c1 INTEGER, c2 CLOB,
                    c3 CLOB NOT NULL);
```

```
ALTER TABLE lvtab ADD c4 CLOB;
```

You can create a WORD (WD) index on a LONG VARCHAR column. You cannot construct other non-FP index types and join indexes on a LONG VARCHAR column.

You can modify a LONG VARCHAR column using the UPDATE, INSERT...VALUES, INSERT...SELECT, LOAD TABLE, DELETE, TRUNCATE, SELECT...INTO and INSERT...LOCATION SQL statements. Positioned updates and deletes are not supported on LONG VARCHAR columns.

You can insert an SAP ASE TEXT column into a LONG VARCHAR column using the INSERT...LOCATION command. All TEXT data inserted is silently right-truncated at 2147483648 bytes (2GB).

7.2.1 LONG VARCHAR Data Type Conversion

There are limited implicit data type conversions to and from the LONG VARCHAR data type and non-LONG VARCHAR data types.

There are no implicit data type conversions from the LONG VARCHAR data type to another non-LONG VARCHAR data type, except LONG BINARY, and CHAR and VARCHAR for INSERT and UPDATE only. There are implicit conversions to LONG VARCHAR data type from CHAR and VARCHAR data types. There are no implicit conversions from BIT, REAL, DOUBLE, NUMERIC, TINYINT, SMALLINT, INT, UNSIGNED INT, BIGINT, UNSIGNED BIGINT, BINARY, VARBINARY, or LONG BINARY data types to LONG VARCHAR data type. Implicit conversion can be controlled using the CONVERSION_MODE database option.

The currently supported string functions for the LONG VARCHAR data type are accepted as input for implicit conversion for the INSERT and UPDATE statements.

The LONG VARCHAR data type can be explicitly converted to CHAR and VARCHAR. No other explicit data type conversions (for example, using the CAST or CONVERT function) exist either to or from the LONG VARCHAR data type.

Truncation of LONG VARCHAR data during conversion of LONG VARCHAR to CHAR is handled the same way the truncation of CHAR data is handled. If the STRING_RTRUNCATION option is ON and string right-truncation of non-spaces occurs, a truncation error is reported and the transaction is rolled back. Trailing partial multibyte characters are replaced with spaces on conversion.

Truncation of LONG VARCHAR data during conversion of LONG VARCHAR to VARCHAR is handled the same way the truncation of VARCHAR data is handled. If the STRING_RTRUNCATION option is ON and string right-truncation of non-spaces occurs, a truncation error is reported and the transaction is rolled back. Trailing partial multibyte characters are truncated on conversion.

Related Information

Function Support [page 93]

7.3 Large Object Variables

SAP IQ supports large object variables.

Inbound LONG BINARY and LONG VARCHAR variables (host variables or SQL variables used by IQ) have no maximum length.

Outbound LONG BINARY and LONG VARCHAR variables (variables set by IQ) have a maximum length of 2GB - 1.

The LOAD TABLE, INSERT…VALUES, INSERT…SELECT, INSERT…LOCATION, SELECT…INTO, and UPDATE SQL statements accept LONG BINARY and LONG VARCHAR variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.

The BIT_LENGTH, BYTE_LENGTH, BYTE_LENGTH64, BYTE_SUBSTR, BYTE_SUBSTR64, CHARINDEX, LOCATE, OCTET_LENGTH, and SUBSTRING64 functions support LONG BINARY and LONG VARCHAR variables of any size data that the SQL variable can hold. In addition, the CHAR_LENGTH, CHAR_LENGTH64, PATINDEX, SUBSTR, and SUBSTRING functions support LONG VARCHAR variables of any size data that the SQL variable can hold.

7.3.1 Large Object Variable Data Type Conversion

The database option ENABLE_LOB_VARIABLES controls the data type conversion of large object variables.

7.3.1.1 ENABLE_LOB_VARIABLES Option

Controls the data type conversion of large object variables.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. At the database level, the value becomes the default for any new user, but has no impact on existing users. At the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Remarks

ENABLE_LOB_VARIABLES controls the data type conversion of large object variables.

When ENABLE_LOB_VARIABLES is OFF, large object variables less than 32K are implicitly converted; an error is reported if a large object variable is greater than or equal to 32K. A LONG VARCHAR variable is implicitly converted to a VARCHAR data type and truncated at 32K. A LONG BINARY variable is implicitly converted to a VARBINARY data type and truncated at 32K.

When ENABLE_LOB_VARIABLES is ON, large object variables of any size retain their original data type and size.

Example

Retain the data type and size of large object variables greater than 32K:

```
SET TEMPORARY OPTION ENABLE_LOB_VARIABLES = ON
```

7.4 Index Support of Large Object Columns

SAP IQ supports the TEXT index on LONG BINARY and LONG VARCHAR columns, and the WORD (WD) index on LONG VARCHAR columns.

7.4.1 TEXT Index Support of Large Object Columns

The TEXT index supports LONG BINARY and LONG VARCHAR columns.

Related Information

SQL Statement Support [page 82]
TEXT Indexes and Text Configuration Objects [page 7]

7.4.2 WD Index Support of LONG VARCHAR (CLOB) Columns

SAP IQ offers limited support for the `WORD` (`WD`) index on `LONG VARCHAR` (`CLOB`) columns.

- The widest column supported by the `WD` index is the maximum width for a LOB column. (The maximum length is equal to 4GB multiplied by the database page size.) The maximum word length supported by SAP IQ is 255 bytes.
- All `sp_iqcheckdb` options for `WD` indexes over `CHAR` and `VARCHAR` columns are also supported for `LONG VARCHAR` (`CLOB`) columns, including allocation, check, and verify modes.
- You can use `sp_iqrebuildindex` to rebuild a `WD` index over a `LONG VARCHAR` (`CLOB`) column.

Chinese text or documents in a binary format require ETL preprocessing to locate and transform words into a form that can be parsed by the `WD` index.

8 SQL Statement Support

Learn about the SQL statements and syntax that support working with `TEXT` indexes and text configurations.

8.1 ALTER TEXT CONFIGURATION Statement

Alters a text configuration object.

> **Note**
>
> This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Quick Links:

Go to Parameters [page 82]

Go to Examples [page 83]

Go to Usage [page 84]

Go to Permissions [page 84]

Syntax

```
ALTER TEXT CONFIGURATION [  <owner>.]<config-name>
   STOPLIST <stoplist>
   | DROP STOPLIST
   | { MINIMUM | MAXIMUM } TERM LENGTH <integer>
   |  TERM BREAKER
      { GENERIC
        [ EXTERNAL NAME  <library-and-entry-point-name-string> ]
       | NGRAM }
   | PREFILTER EXTERNAL NAME <library-and-entry-point-name-string>
   | DROP PREFILTER
```

Parameters

(back to top) [page 82]

- **stoplist** a string-expression used to create or replace the list of terms to ignore when building a TEXT index. Terms specified in this list are also ignored in a query. Separate stoplist terms with spaces. Stoplist terms cannot contain whitespace and should not contain non-alphanumeric characters. Non-

alphanumeric characters are interpreted as spaces and break the term into multiple terms. For example, “and/or” is interpreted as the two terms “and” and “or”. The maximum number of stoplist terms is 7999.

- **DROP STOPLIST** use to drop the stoplist for a text configuration object.
- **MINIMUM TERM LENGTH** specifies the minimum length, in characters, of a term to include in the TEXT index. The value specified in the MINIMUM TERM LENGTH clause is ignored when using NGRAM TEXT indexes. Terms that are shorter than this setting are ignored when building or refreshing the TEXT index. The value of this option must be greater than 0. If you set this option to be higher than MAXIMUM TERM LENGTH, the value of MAXIMUM TERM LENGTH is automatically adjusted to be the same as the new MINIMUM TERM LENGTH value.
- **MAXIMUM TERM LENGTH** with GENERIC TEXT indexes, specifies the maximum length, in characters, of a term to include in the TEXT index. Terms that are longer than this setting are ignored when building or refreshing the TEXT index. The value of MAXIMUM TERM LENGTH must be less than or equal to 60. If you set this option to be lower than MINIMUM TERM LENGTH, the value of MINIMUM TERM LENGTH is automatically adjusted to be the same as the new MAXIMUM TERM LENGTH value.
- **TERM BREAKER** specifies the name of the algorithm to use for separating column values into terms. The choices for IN SYSTEM tables are GENERIC (the default) or NGRAM. The GENERIC algorithm treats any string of one or more alphanumerics, separated by non-alphanumerics, as a term. The NGRAM algorithm breaks strings into n-grams. An n-gram is an n-character substring of a larger string. The NGRAM term breaker is required for fuzzy (approximate) matching, or for documents that do not use whitespace or non-alphanumeric characters to separate terms. NGRAM is supported for IN SYSTEM tables.

  NGRAM term breaker is built on TEXT indexes, so use text configuration object settings to define whether to use an NGRAM or GENERIC TEXT index.

  TERM BREAKER can include the specification for the external term breaker library using EXTERNAL NAME and the library entry point.
- **library-and-entry-point-name-string** `<[operating-system:]function-name@library>`
- **PREFILTER EXTERNAL NAME** specifies the entry_point and the library name of the external pre-filter library provided by external vendors.
- **DROP PREFILTER** drops the external prefilter and sets NULL to the prefilter columns in `ISYSTEXTCONFIG` table.

Examples

(back to top) [page 82]

Example 1

creates a text configuration object, `maxTerm16`, and then change the maximum term length to 16:

```
CREATE TEXT CONFIGURATION maxTerm16 FROM default_char;
ALTER TEXT CONFIGURATION maxTerm16 MAXIMUM TERM LENGTH 16;
```

Example 2

adds stoplist terms to the `maxTerm16` configuration object:

```
ALTER TEXT CONFIGURATION maxTerm16
STOPLIST 'because about therefore only';
```

Example 3

updates the text configuration object, `my_text_config`, to use the entry point `my_term_breaker` in the external library `mytermbreaker.dll` for breaking the text:

```
CREATE TEXT CONFIGURATION my_text_config FROM default_char;
ALTER TEXT CONFIGURATION my_text_config
TERM BREAKER GENERIC EXTERNAL NAME 'platform:my_term_breaker@mytermbreaker';
```

Example 4

updates the text configuration object, `my_text_config`, to use the entry point `my_prefilter` in the external library `myprefilter.dll` for prefiltering the documents:

```
ALTER TEXT CONFIGURATION my_text_config
PREFILTER EXTERNAL NAME 'platform:my_prefilter@myprefilter';
```

Usage

(back to top) [page 82]

TEXT indexes are dependent on a text configuration object. SAP IQ TEXT indexes use immediate refresh, and cannot be truncated; you must drop the indexes before you can alter the text configuration object. To view the settings for text configuration objects, query the `SYSTEXTCONFIG` system view.

Side Effects:

- Automatic commit.

Permissions

(back to top) [page 82]

`TERM BREAKER` or `PREFILTER EXTERNAL NAME` clause – Requires the CREATE ANY EXTERNAL REFERENCE system privilege, along with one of:

- ALTER ANY TEXT CONFIGURATION system privilege.
- ALTER ANY OBJECT system privilege.
- You own the text configuration object.

All other clauses – Requires the ALTER ANY TEXT CONFIGURATION system privilege, regardless of whether the user is the owner of the configuration object.

8.2 ALTER TEXT INDEX Statement

Renames, moves or alters the definition of a TEXT index.

> **Note**
>
> This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Quick Links:

Go to Parameters [page 85]

Go to Examples [page 85]

Go to Usage [page 86]

Go to Permissions [page 86]

Syntax

```
ALTER TEXT INDEX [<owner.>]<text-index-name>
   ON [<owner.>]<table-name>
   alter-clause [page 85]
alter-clause - (back to Syntax) [page 85]
   rename-object [page 85] | move-object [page 85]
rename-object - (back to alter-clause) [page 85]
   RENAME { AS | TO } <new-name>
move-object - (back to alter-clause) [page 85]
   MOVE TO <dbspace-name>
```

Parameters

(back to top) [page 85]

- **RENAME** rename the TEXT index.
- **MOVE** move the TEXT index to the specified dbspace.

Examples

(back to top) [page 85]

Example

creates a TEXT index, `MyTextIndex`, defining it as IMMEDIATE REFRESH, rename the TEXT index to `Text_index_daily`, and move the TEXT index to a dbspace named `tispace`:

```
CREATE TEXT INDEX MyTextIndex ON Customers ( CompanyName ) IMMEDIATE REFRESH;
```

```
ALTER TEXT INDEX MyTextIndex ON Customers RENAME AS Text_index_daily;
ALTER TEXT INDEX Text_Index_Daily ON Customers MOVE TO tispace;
```

Usage

(back to top) [page 85]

Side Effects:

- Automatic commit.

Permissions

(back to top) [page 85]

`move-object` clause – Requires one of:

- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- REFERENCES privilege the underlying table.
- You own the underlying table.

`rename-object` clause – Requires one of:

- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- MANAGE ANY DBSPACE.
- One of the following:
  - You own the underlying table being indexed.
  - REFERENCES privilege on the table along with one of the following:
    - CREATE ANY OBJECT system privilege.
    - CREATE privilege on the target dbspace.

8.3 CREATE TEXT CONFIGURATION Statement

Creates a text configuration object.

> **Note**
>
> This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Quick Links:

Go to Parameters [page 87]

Go to Examples [page 87]

Go to Usage [page 87]

Go to Permissions [page 88]

Syntax

```
CREATE TEXT CONFIGURATION [ <owner>.]<new-config-name>
   FROM [ <owner>.]<existing-config-name>
```

Parameters

(back to top) [page 86]

- **FROM** specifies the name of a text configuration object to use as the template for creating the new text configuration object. The names of the default text configuration objects are DEFAULT_CHAR and DEFAULT_NCHAR. DEFAULT_CHAR is supported for SAP IQ tables only; DEFAULT_NCHAR is supported on SAP SQL Anywhere tables only.

Examples

(back to top) [page 86]

Example 1

creates a text configuration object, `max_term_sixteen`, using the `default_char` text configuration object, then use `ALTER TEXT CONFIGURATION` to change the maximum term length for `max_term_sixteen` to 16:

```
CREATE TEXT CONFIGURATION max_term_sixteen FROM default_char;
```

```
ALTER TEXT CONFIGURATION max_term_sixteen MAXIMUM TERM LENGTH 16;
```

Usage

(back to top) [page 86]

Create a text configuration object using another text configuration object as a template, then alter the options as needed using the `ALTER TEXT CONFIGURATION` statement.

To view the list of all text configuration objects and their settings in the database, query the `SYSTEXTCONFIG` system view.

Side Effects:

- Automatic commit.

Permissions

(back to top) [page 86]

Text configuration object to be owned by self –

- Requires CREATE TEXT CONFIGURATION system privilege.

Text configuration object to be owned by any user – Requires one of:

- CREATE ANY TEXT CONFIGURATION system privilege.
- CREATE ANY OBJECT system privilege.

All text configuration objects have PUBLIC access. Any user with privilege to create a `TEXT` index can use any text configuration object.

8.4 CREATE TEXT INDEX Statement

Creates a TEXT index and specifies the text configuration object to use.

> **Note**
>
> This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Quick Links:

Go to Parameters [page 89]

Go to Examples [page 89]

Go to Usage [page 89]

Go to Permissions [page 89]

Syntax

```
CREATE TEXT INDEX <text-index-name>
   ON [ <owner.>]<table-name>( <column-name>, ...)
   [ IN <dbspace-name> ]
   [ CONFIGURATION [ <owner.>]<text-configuration-name>]
   [ IMMEDIATE REFRESH ]
```

Parameters

(back to top) [page 88]

- **ON** specifies the table and column on which to build the TEXT index.
- **IN** specifies the dbspace in which the TEXT index is located. If this clause is not specified, then the TEXT index is created in the same dbspace as the underlying table.
- **CONFIGURATION** specifies the text configuration object to use when creating the TEXT index. If this clause is not specified, the default_char text configuration object is used.
- **IMMEDIATE REFRESH** (default) refreshes the TEXT index each time changes in the underlying table impact data in the TEXT index. Only permitted value for tables in SAP IQ main store. Once created, the IMMEDIATE REFRESH clause cannot be changed.

Examples

(back to top) [page 88]

Example 1

creates a TEXT index, `myTxtIdx`, on the `CompanyName` column of the `Customers` table in the `iqdemo` database, using the `max_term_sixteen` text configuration object:

```
CREATE TEXT INDEX myTxtIdx ON Customers (CompanyName );
```

```
CONFIGURATION max_term_sixteen;
```

Usage

(back to top) [page 88]

You cannot create a TEXT index on views or temporary tables, or on an IN SYSTEM materialized view. The `BEGIN PARALLEL IQ…END PARALLEL IQ` statement does not support `CREATE TEXT INDEX`.

Side Effects:

- Automatic commit.

Permissions

(back to top) [page 88]

Requites one of:

- CREATE ANY INDEX system privilege along with CREATE privilege on the dbspace where the index is being created.

- CREATE ANY OBJECT system privilege.

8.5 DROP TEXT CONFIGURATION Statement

Drops a text configuration object.

> **Note**
>
> This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Quick Links:

Go to Examples [page 90]

Go to Usage [page 90]

Go to Permissions [page 91]

Syntax

```
DROP TEXT CONFIGURATION  [ <owner>.]<text-config-name>
```

Examples

(back to top) [page 90]

Example 1

creates and drops the `mytextconfig` text configuration object:

```
CREATE TEXT CONFIGURATION mytextconfig FROM default_char;
DROP TEXT CONFIGURATION mytextconfig;
```

Usage

(back to top) [page 90]

Use `DROP TEXT CONFIGURATION` to drop a text configuration object.

Attempting to drop a text configuration object with dependent TEXT indexes results in an error. You must drop the dependent TEXT indexes before dropping the text configuration object.

Text configuration objects are stored in the `ISYSTEXTCONFIG` system table.

Side Effects:

- Automatic commit.

Permissions

(back to top) [page 90]

Text configuration object owned by self – None required.

Table configuration object owned by any user – Requires one of:

- DROP ANY TEXT CONFIGURATION system privilege.
- DROP ANY OBJECT system privilege.

8.6 DROP TEXT INDEX Statement

Removes a TEXT index from the database.

> **Note**
>
> This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Quick Links:

Go to Parameters [page 91]

Go to Examples [page 92]

Go to Usage [page 92]

Go to Permissions [page 92]

Syntax

```
DROP TEXT INDEX <text-index-name>
   ON [ <owner> ] <table-name>
```

Parameters

(back to top) [page 91]

- **ON** specifies the table on which the TEXT index is built.

Examples

(back to top) [page 91]

Example 1

creates and drops the `TextIdx` TEXT index:

```
CREATE TEXT INDEX TextIdx ON Customers ( Street );
DROP TEXT INDEX TextIdx ON Customers;
```

Usage

(back to top) [page 91]

You must drop dependent TEXT indexes before you can drop a text configuration object.

Side Effects:

- Automatic commit.

Permissions

(back to top) [page 91]

Requires one of:

- DROP ANY INDEX system privilege.
- DROP ANY OBJECT system privilege.
- REFERENCES privilege on the table being indexed.
- You own the underlying table.

9 Function Support

Learn about the functions that support the `LONG BINARY` and `LONG VARCHAR` data types.

9.1 Summary of Function Support of Large Object Data

A summary of function support of `LONG BINARY (BLOB)` and `LONG VARCHAR (CLOB)` data types and `LONG BINARY` and `LONG VARCHAR` variables.

In addition to the functions listed in this table, you can use the `BFILE` function to extract LOB data.

Scalar and aggregate user-defined functions support large object data types as input parameters. .

Table 26: Function Support of LOB Data Types and Variables

Function	BLOB data supported?	BLOB variables supported?	CLOB data supported?	CLOB variables supported?
BIT_LENGTH()	Yes	Yes	Yes	Yes
BYTE_LENGTH()	Yes*	Yes*	Yes*	Yes*
BYTE_LENGTH64()	Yes	Yes	Yes	Yes
BYTE_SUBSTR()	Yes	Yes	Yes	Yes
BYTE_SUBSTR64()	Yes	Yes	Yes	Yes
CHAR_LENGTH()	No	No	Yes	Yes
CHAR_LENGTH64()	No	No	Yes	Yes
CHARINDEX()	Yes	Yes	Yes	Yes
LOCATE()	Yes	Yes	Yes	Yes
OCTET_LENGTH()	Yes	Yes	Yes	Yes
PATINDEX()	No	No	Yes	Yes
SUBSTR() / SUBSTRING()	No	No	Yes	Yes
SUBSTRING64()	Yes	Yes	Yes	Yes

*The `BYTE_LENGTH` function supports both `LONG BINARY` columns and variables and `LONG VARCHAR` columns and variables, only if the query returns less than 2GB. If the byte length of the returned `LONG BINARY` or `LONG VARCHAR` data is greater than 2GB, `BYTE_LENGTH` returns an error that says you must use the `BYTE_LENGTH64` function.

Related Information

User-Defined Function Support of Large Object Columns [page 105]
Large Object Data Exports [page 70]

9.2 BIT_LENGTH Function

The `BIT_LENGTH` function returns an unsigned 64-bit value containing the bit length of the large object column or variable parameter. If the argument is NULL, `BIT_LENGTH` returns NULL.

Syntax

```
BIT_LENGTH( <large-object-column> )
```

Parameters

`<large-object-column>` – the name of a `LONG VARCHAR` or `LONG BINARY` column or variable.

Usage

`BIT_LENGTH` supports all SAP IQ data types and `LONG BINARY` and `LONG VARCHAR` variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.

9.3 BYTE_LENGTH Function [String]

Returns the number of bytes in a string.

Syntax

```
BYTE_LENGTH ( <string-expression> )
```

Parameters

Table 27:

Parameters	Description
string-expression	The string whose length is to be calculated.

Returns

INT

Remarks

Trailing white space characters are included in the length returned.

The return value of a NULL string is NULL.

If the string is in a multibyte character set, the `BYTE_LENGTH` value differs from the number of characters returned by `CHAR_LENGTH`.

If you are licensed to use the Unstructured Data Analytics functionality, you can use this function with large object data:

- The `BYTE_LENGTH` function supports both `LONG BINARY` columns and variables and `LONG VARCHAR` columns and variables, only if the query returns less than 2GB. If the byte length of the returned `LONG BINARY` or `LONG VARCHAR` data is greater than or equal to 2GB, `BYTE_LENGTH` returns an error that says you must use the `BYTE_LENGTH64` function.

Standards and Compatibility

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- SAP Database Products—Not supported by SAP ASE.

Example

Returns the value 12:

```
SELECT BYTE_LENGTH( 'Test Message' ) FROM iq_dummy
```

9.4 BYTE_LENGTH64 Function [String]

The BYTE_LENGTH64 function returns an unsigned 64-bit value containing the byte length of the large object column or variable parameter.

Syntax

```
BYTE_LENGTH64(  <large-object-column> )
```

Parameters

<large-object-column> – the name of a LONG VARCHAR or LONG BINARY column or variable.

Usage

The BYTE_LENGTH64 function supports both LONG BINARY and LONG VARCHAR columns and LONG BINARY and LONG VARCHAR variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.

9.5 BYTE_SUBSTR64 and BYTE_SUBSTR Functions [String]

The BYTE_SUBSTR64 and BYTE_SUBSTR functions return the byte substring of the large object column or variable parameter.

Syntax

```
BYTE_SUBSTR64( <large-object-column>, <start>, <length> )
```

```
BYTE_SUBSTR( <large-object-column>, <start>, <length> )
```

Parameters

<large-object-column> – the name of a LONG VARCHAR or LONG BINARY column or variable.

<start> – an integer expression indicating the start of the substring. A positive integer starts from the beginning of the string, with the first byte at position 1. A negative integer specifies a substring starting from the end of the string, with the final byte at position -1.

<length> – an integer expression indicating the length of the substring. A positive length specifies the number of bytes to return, starting at the <start> position. A negative length specifies the number of bytes to return, ending at the <start> position.

Usage

- Nested operations of the functions BYTE_LENGTH64, BYTE_SUBSTR64, and BYTE_SUBSTR do not support large object columns or variables.
- The BYTE_SUBSTR64 and BYTE_SUBSTR functions support both LONG BINARY and LONG VARCHAR columns and LONG BINARY and LONG VARCHAR variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.

9.6 CHAR_LENGTH Function [String]

The CHAR_LENGTH function returns a signed 32-bit value containing the character length of the LONG VARCHAR column or variable parameter, including the trailing blanks.

Syntax

```
CHAR_LENGTH( <long-varchar-object> )
```

Parameters

<long-varchar-object> – the name of a LONG VARCHAR column or LONG VARCHAR variable.

Usage

- CHAR_LENGTH supports LONG VARCHAR columns and LONG VARCHAR variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.
- If the argument is NULL, CHAR_LENGTH returns NULL.
- If the character length exceeds 2GB - 1 (2147483647), an error is returned.

9.7 CHAR_LENGTH64 Function [String]

The CHAR_LENGTH64 function returns an unsigned 64-bit value containing the character length of the LONG VARCHAR column or variable parameter, including the trailing blanks.

Syntax

```
CHAR_LENGTH64( <long-varchar-object> )
```

Parameters

<long-varchar-object> – the name of a LONG VARCHAR column in a table or a LONG VARCHAR variable.

Usage

- CHAR_LENGTH64 supports LONG VARCHAR columns and LONG VARCHAR variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.
- If the argument is NULL, CHAR_LENGTH64 returns NULL.

9.8 CHARINDEX Function [String]

The CHARINDEX function returns a 64-bit signed integer containing the position of the first occurrence of the specified string in the large object column or variable parameter. For CHAR and VARCHAR columns, CHARINDEX returns a 32-bit signed integer position.

Syntax

```
CHARINDEX( <string-expression>, <large -object-column> )
```

Parameters

`<string-expression>` – the string of up to 255 bytes, for which you are searching.

`<large-object-column>` – the name of the `LONG VARCHAR` or `LONG BINARY` column or variable.

Usage

- All the positions or offsets, returned or specified, in the `CHARINDEX` function are always character offsets and may be different from the byte offset for multibyte data.
- If the large object cell being searched contains more than one instance of `<string-expression>`, `CHARINDEX` returns only the position of the first instance.
- If the column does not contain the string, the `CHARINDEX` function returns zero (0).
- Searching for a string longer than 255 bytes returns NULL.
- Searching for a zero-length string returns 1.
- If any of the arguments is NULL, the result is NULL.
- `CHARINDEX` supports searching `LONG VARCHAR` and `LONG BINARY` columns and `LONG VARCHAR` and `LONG BINARY` variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.

9.9 LOCATE Function [String]

The `LOCATE` function returns a 64-bit signed integer containing the position of the specified string in the large object column or variable parameter. For `CHAR` and `VARCHAR` columns, `LOCATE` returns a 32-bit signed integer position.

Syntax

```
LOCATE( <large-object-column>, <string-expression>
[, <numeric-expression> ] )
```

Parameters

<large-object-column> – the name of the LONG VARCHAR or LONG BINARY column or variable to search.

<string-expression> – the string of up to 255 bytes, for which you are searching.

<numeric-expression> – the character position or offset at which to begin the search in the string. The <numeric-expression> is a 64-bit signed integer for LONG VARCHAR and LONG BINARY columns and is a 32-bit signed integer for CHAR, VARCHAR, and BINARY columns. The first character is position 1. If the starting offset is negative, LOCATE returns the last matching string offset, rather than the first. A negative offset indicates how much of the end of the string to exclude from the search. The number of characters excluded is calculated as (-1 * offset) - 1.

Usage

- All the positions or offsets, returned or specified, in the LOCATE function are always character offsets and may be different from the byte offset for multibyte data.
- If the large object cell being searched contains more than one instance of the string:
  - If <numeric-expression> is specified, LOCATE starts the search at that offset in the string.
  - If <numeric-expression> is not specified, LOCATE returns only the position of the first instance.
- If the column does not contain the string, LOCATE returns zero (0).
- Searching for a string longer than 255 bytes returns NULL.
- Searching for a zero-length string returns 1.
- If any of the arguments is NULL, the result is NULL.
- LOCATE supports searching LONG VARCHAR and LONG BINARY columns and LONG VARCHAR and LONG BINARY variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.

9.10 OCTET_LENGTH Function [String]

The OCTET_LENGTH function returns an unsigned 64-bit value containing the byte length of the large object column or variable parameter.

Syntax

```
OCTET_LENGTH( <column-name> )
```

Parameters

<large-object-column> – the name of a LONG VARCHAR or LONG BINARY column or variable.

Usage

- If the argument is NULL, OCTET_LENGTH returns NULL.
- OCTET_LENGTH supports all SAP IQ data types and LONG VARCHAR and LONG BINARY variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.

9.11 PATINDEX Function [String]

The PATINDEX function returns a 64-bit unsigned integer containing the position of the first occurrence of the specified pattern in a LONG VARCHAR column or variable. For CHAR and VARCHAR columns, PATINDEX returns a 32-bit unsigned integer position.

Syntax

```
PATINDEX( '<%pattern%>', <long-varchar-column> )
```

Parameters

pattern the pattern for which you are searching. This string is limited to 126 bytes for patterns with wildcards. If you omit the leading percent wildcard, PATINDEX returns one (1) if the pattern occurs at the beginning of the column value, and zero (0) if the pattern does not occur at the beginning of the column value. Similarly, if you omit the trailing percent wildcard, the pattern should occur at the end of the column value. The pattern uses the same wildcards as the LIKE comparison.

Patterns without wildcards—percent (%) and underscore (_)— can be up to 255 bytes in length.

long-varchar-column the name of the LONG VARCHAR column or variable.

Usage

- All the positions or offsets, returned or specified, in the PATINDEX function are always character offsets and may be different from the byte offset for multibyte data.

- If the LONG VARCHAR cell being searched contains more than one instance of the string pattern, PATINDEX returns only the position of the first instance.
- If the column does not contain the pattern, PATINDEX returns zero (0).
- Searching for a pattern longer than 126 bytes returns NULL.
- Searching for a zero-length pattern returns 1.
- If any of the arguments is NULL, the result is zero (0).
- PATINDEX supports LONG VARCHAR variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length. PATINDEX does not support LONG BINARY variables or searching LONG BINARY columns.

9.12 SUBSTRING Function [String]

The SUBSTRING function returns a variable-length character string of the LONG VARCHAR column or variable parameter. If any of the arguments are NULL, SUBSTRING returns NULL.

Syntax

```
{ SUBSTRING |  SUBSTR } ( <long-varchar-column>, <start> [, <length> ] )
```

Parameters

<long-varchar-column> – the name of a LONG VARCHAR column or variable.

<start> – an integer expression indicating the start of the substring. A positive integer starts from the beginning of the string, with the first character at position 1. A negative integer specifies a substring starting from the end of the string, with the final character at position -1.

<length> – an integer expression indicating the character length of the substring. A positive length specifies the number of characters to return, starting at the <start> position. A negative length specifies the number of characters to return, ending at the <start> position.

Usage

SUBSTRING supports LONG VARCHAR variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length. SUBSTRING does not support LONG BINARY variables or searching LONG BINARY columns.

When the ansi_substring database option is set to ON (default), negative values are invalid.

9.12.1 ANSI_SUBSTRING Option [TSQL]

Controls the behavior of the SUBSTRING (SUBSTR) function when negative values are provided for the start or length parameters.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. At the database level, the value becomes the default for any new user, but has no impact on existing users. At the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Remarks

When the `ANSI_SUBSTRING` option is set to ON, the behavior of the `SUBSTRING` function corresponds to ANSI/ISO SQL/2003 behavior. A negative or zero start offset is treated as if the string were padded on the left with noncharacters, and gives an error if a negative length is provided.

When this option is set to OFF, the behavior of the `SUBSTRING` function is the same as in earlier versions of SAP IQ: a negative start offset means an offset from the end of the string, and a negative length means the desired substring ends length characters to the left of the starting offset. Using a start offset of 0 is equivalent to a start offset of 1.

Avoid using nonpositive start offsets or negative lengths with the `SUBSTRING` function. Where possible, use the `LEFT` or `RIGHT` functions instead.

Example

These examples show the difference in the values returned by the SUBSTRING function based on the setting of the ANSI_SUBSTRING option:

```
SUBSTRING( 'abcdefgh',-2,4 );
    ansi_substring = Off ==> 'gh'
    // substring starts at second-last character
    ansi_substring = On  ==> 'gh'
    // takes the first 4 characters of
    // ???abcdefgh and discards all ?
```

```
SUBSTRING( 'abcdefgh',4,-2 );
    ansi_substring = Off ==> 'cd'
    ansi_substring = On  ==> value -2 out of range
    for destination
```

```
SUBSTRING( 'abcdefgh',0,4 );
    ansi_substring = Off ==> 'abcd'
    ansi_substring = On  ==> 'abcd'
```

9.13 SUBSTRING64 Function [String]

The SUBSTRING64 function returns a variable-length character string of the large object column or variable parameter.

Syntax

```
SUBSTRING64 ( <large-object-column>, <start> [, <length> ] )
```

Parameters

<large-object-column> – the name of a LONG VARCHAR or LONG BINARY column or variable.

<start> – an 8-byte integer indicating the start of the substring. SUBSTRING64 interprets a negative or zero <start> offset as if the string were padded on the left with "non-characters." The first character starts at position 1.

<length> – an 8-byte integer indicating the length of the substring. If <length> is negative, an error is returned.

Example

Values returned by `SUBSTRING64`, given a column named `col1` that contains the string ("ABCDEFG"):

`SUBSTRING64( col1, 2, 4 )` returns the string "BCDE"

`SUBSTRING64( col1, 1, 3 )` returns the string "ABC"

`SUBSTRING64( col1, 0, 3 )` returns the string "AB"

`SUBSTRING64( col1, -1, 3 )` returns the string "A"

Usage

- If any of the arguments are NULL, `SUBSTRING64` returns NULL.
- Nested operations of the functions `SUBSTRING64`, `SUBSTRING`, `SUBSTR`, `BYTE_SUBSTR`, and `BYTE_SUBSTR64` do not support large object columns or variables.
- `SUBSTRING64` supports searching `LONG VARCHAR` and `LONG BINARY` columns and `LONG VARCHAR` and `LONG BINARY` variables of any size of data. Currently, a SQL variable can hold up to 2GB - 1 in length.

9.14 Aggregate Function Support of Large Object Columns

Only the aggregate function `COUNT (*)` is supported for `LONG BINARY` and `LONG VARCHAR` columns.

The `COUNT DISTINCT` parameter is not supported. An error is returned if a `LONG BINARY` or `LONG VARCHAR` column is used with the `MIN`, `MAX`, `AVG`, or `SUM` aggregate functions.

9.15 User-Defined Function Support of Large Object Columns

Scalar and aggregate user-defined functions support large object data types `LONG VARCHAR` (`CLOB`) and `LONG BINARY` (`BLOB`) up to 4GB (gigabytes) as input parameters. LOB data types are not supported as output parameters.

You must be specifically licensed to use the User-defined Function support functionality. .

Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

Accessibility

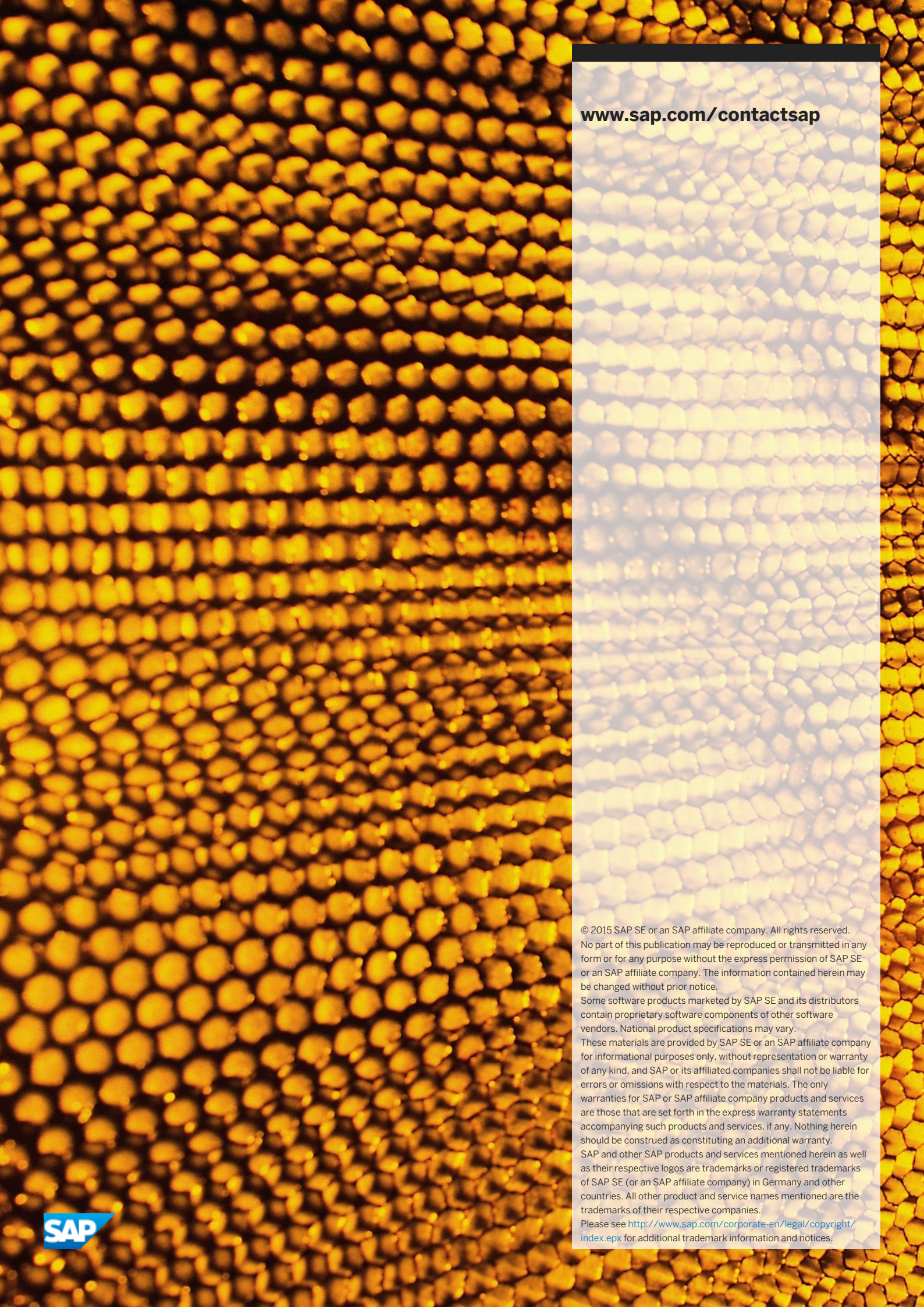
The information contained in the SAP documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of wilful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: http://help.sap.com/disclaimer).



www.sap.com/contactsap