SYBASE®

Unstructured Data Analytics in Sybase IQ

Sybase IQ

15.2

DOCUMENT ID: DC01268-01-1520-01

LAST REVISED: April 2010

Contents

About This Book

Audience

This book is for Sybase® IQ users who require reference material for working with unstructured data in Sybase IQ. This manual is the place to look for information such as available syntax, parameters, functions, stored procedures, indexes, and options related to unstructured data analytics features in Sybase IQ. Use this book as a reference together with the other books in the Sybase IQ documentation set to understand storage and retrieval of unstructured data within the Sybase IQ database.

Related Sybase IQ documents

The Sybase IQ 15.2 documentation set includes:

- *Release Bulletin* for your platform – contains last-minute information that was too late to be included in the books.

  A more recent version of the release bulletin may be available. To check for critical product or document information that was added after the release of the product CD, use the Sybase Product Manuals Web site.

- *Installation and Configuration Guide* for your platform – describes installation, upgrading, and some configuration procedures for Sybase IQ.
- *New Features Summary Sybase IQ 15.2* – summarizes new features and behavior changes for the current version.
- *Advanced Security in Sybase IQ* – covers the use of user-encrypted columns within the Sybase IQ data repository. You need a separate license to install this product option.
- *Error Messages* lists Sybase IQ – error messages referenced by Sybase error code, SQLCode, and SQLState, and SQL preprocessor errors and warnings.
- *IMSL Numerical Library User's Guide: Volume 2 of 2 C Stat Library* – contains a concise description of the IMSL C Stat Library time series C functions. This book is available only to RAP – The Trading Edition® Enterprise users.

- *Introduction to Sybase IQ* – includes exercises for those unfamiliar with Sybase IQ or with the Sybase Central™ database management tool.
- *Performance and Tuning Guide* – describes query optimization, design, and tuning issues for very large databases.
- *Quick Start* – discusses how to build and query the demo database provided with Sybase IQ for validating the Sybase IQ software installation. Includes information on converting the demo database to multiplex.
- *Reference Manual* – reference guides to Sybase IQ:
  - *Reference: Building Blocks, Tables, and Procedures* – describes SQL, stored procedures, data types, and system tables that Sybase IQ supports.
  - *Reference: Statements and Options* – describes the SQL statements and options that Sybase IQ supports.
- *System Administration Guide* – includes:
  - *System Administration Guide: Volume 1* – describes start-up, connections, database creation, population and indexing, versioning, collations, system backup and recovery, troubleshooting, and database repair.
  - *System Administration Guide: Volume 2* – describes how to write and run procedures and batches, program with OLAP, access remote data, and set up IQ as an Open Server™. This book also discusses scheduling and event handling, XML programming, and debugging.
- *Time Series Guide* – describes SQL functions used for time series forecasting and analysis. You need RAP – The Trading Edition™ Enterprise to use this product option.
- *Unstructured Data Analytics in Sybase IQ* – explains how to store and retrieve unstructured data in Sybase IQ databases. You need a separate license to install this product option.
- *User-Defined Functions Guide* – provides information about user-defined functions, their parameters, and possible usage scenarios.
- *Using Sybase IQ Multiplex* – tells how to use multiplex capability, which manages large query loads across multiple nodes.
- *Utility Guide* – provides Sybase IQ utility program reference material, such as available syntax, parameters, and options.

The Sybase IQ 15.2 documentation set is available online at Product Manuals at http://sybooks.sybase.com.

Related SQL Anywhere documentation

Because Sybase IQ shares many components with SQL Anywhere Server, a component of the SQL Anywhere® package, Sybase IQ supports many of the same features as SQL Anywhere Server. The IQ documentation set refers you to SQL Anywhere documentation, where appropriate.

Documentation for SQL Anywhere includes:

- *SQL Anywhere Server – Database Administration* describes how to run, manage, and configure SQL Anywhere databases. It describes database connections, the database server, database files, backup procedures, security, high availability, and replication with Replication Server®, as well as administration utilities and options.
- *SQL Anywhere Server – Programming* describes how to build and deploy database applications using the C, C++, Java, PHP, Perl, Python, and .NET programming languages such as Visual Basic and Visual C#. This book also describes a variety of programming interfaces, such as ADO.NET and ODBC.
- *SQL Anywhere Server – SQL Reference* provides reference information for system procedures, and the catalog (system tables and views). It also provides an explanation of the SQL Anywhere implementation of the SQL language (search conditions, syntax, data types, and functions).
- *SQL Anywhere Server – SQL Usage* describes how to design and create databases; how to import, export, and modify data; how to retrieve data; and how to build stored procedures and triggers.

You can also refer to the SQL Anywhere documentation in the SQL Anywhere 11.0.1 collection in DocCommentXchange at http://dcx.sybase.com/dcx_home.php and at the Sybase Product Manuals Web site.

CHAPTER 1

Introduction to Unstructured Data Analytics in Sybase IQ

This chapter introduces you to unstructured data analytics in Sybase IQ and describes the compatibility and conformance to standards of Sybase IQ large object data.

The Unstructured Data Analytics Option

The Unstructured Data Analytics Option extends the capabilities of Sybase IQ to allow storage, retrieval, and full text searching of binary large objects (BLOBs) and character large objects (CLOBs) within the Sybase IQ 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. Sybase 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, Sybase IQ lets organizations access both types of data using the same application and the same interface. The full text search capability of Sybase IQ supports text archival applications (text analytics) in handling unstructured and semistructured data.

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.

Compatibility

SQL Anywhere (SA) 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 SQL Anywhere is SQL/2003 compliant. SQL Anywhere does not support the BYTE_LENGTH64, BYTE_SUBSTR64, BFILE, BIT_LENGTH, OCTET_LENGTH, CHAR_LENGTH64, and SUBSTRING64 functions.

Adaptive Server® Enterprise (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 Adaptive Server Enterprise is an ANSI SQL Transact-SQL® extension.

Conformance to standards

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

CHAPTER 2

TEXT Indexes and Text Configuration Objects

This chapter describes 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.

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.

Comparison of WD and TEXT indexes

This table compares WD and TEXT indexes in terms of syntax and capability.

Table 2-1: 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')`
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%'`	No
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.

Creating a TEXT index

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

❖ Creating a TEXT index (Sybase Central)

1 Connect to the database as a user with DBA or RESOURCE authority.
2 In the left pane, right-click the Text Indexes folder and select New > Text Index.
3 Select the table on which to create the TEXT index.
4 Type a name for the TEXT index. Click Next.
5 Select the column to include in the index. Click Next.
6 Select the text configuration object to use when processing the data for the TEXT index. Click Next.
7 For SQL Anywhere tables, in the Specify a Refresh Type dialog, click Next.

   For Sybase IQ tables, this option does not appear. The only supported refresh type is Immediate.
8 Select the dbspace where the TEXT index will be stored.
9 Click Next.
10 Type a comment describing the text configuration, and click Finish.

❖ Creating a TEXT index (Interactive SQL)

1 Connect to the database as a user with DBA or RESOURCE authority.
2 Execute a CREATE TEXT INDEX statement.

   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
   ```

Guidelines for TEXT index size estimation

To estimate TEXT index main store size, use this formula:

*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.

TEXT index restrictions

Sybase IQ text configuration objects and TEXT indexes have these limitations by design:

- Sybase IQ engine 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.
- The NGRAM term-breaker is not supported for TEXT indexes. Sybase IQ currently only supports the GENERIC term-breaker and one TEXT index per column.

Displaying a list of TEXT indexes

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

❖ Displaying a list of TEXT indexes (Sybase Central)

1. Connect to the database as a user with DBA or RESOURCE authority.
2. In the left pane, select the Text Indexes folder.

   A list of all TEXT indexes appears in the right pane.

❖ Displaying a list of TEXT indexes (Interactive SQL)

1. Connect to the database as a user with DBA or RESOURCE authority.
2. Execute a SELECT statement.

   For example, to list all Sybase IQ TEXT indexes, enter:

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

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

```
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;
```

Editing a TEXT index

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

❖ Editing a TEXT index (Sybase Central)

1. Connect to the database as a user with DBA or RESOURCE authority.
2. In the left pane, select the Text Indexes folder.
3. In the list of Text Indexes, right-click the object to modify and select Properties.
4. On the General tab, modify the settings as needed.
5. Click OK.

❖ **Editing a TEXT index (Interactive SQL)**

1. Connect to the database as a user with DBA or RESOURCE authority.
2. Execute an ALTER TEXT INDEX statement.

   For example, to rename the myTxtIdx to MyTextIndex, enter:

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

Modifying the location

Change the dbspace where the TEXT index is stored.

❖ **Modifying the location (Sybase Central)**

1. Connect to the database as a user with DBA or SPACE ADMIN authority or as table owner with CREATE privilege on dbspace.
2. In the left pane, select the Text Indexes folder.
3. In the list of Text Indexes, right-click the object to modify and select Properties.
4. On the General tab, select the dbspace from the drop-down list.
5. When the dbspace is updated, click OK.

❖ **Modifying the location (Interactive SQL)**

1. Connect to the database as a user with DBA or SPACE ADMIN authority.
2. Execute a ALTER TEXT INDEX statement with the MOVE TO clause

   For example, to move the TEXT index MyTextIndex to a dbspace named tispace, enter:

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

Dropping a TEXT index

Drop TEXT indexes from the database.

❖ **Dropping a TEXT index (Sybase Central)**

1. Connect to the database as a user with DBA or RESOURCE authority.
2. In the left pane, select the Text Indexes folder.
3. In the list of Text Indexes, right-click the object to modify and select Delete.
4. In the confirmation dialog, click Yes.

❖ **Dropping a TEXT index (Interactive SQL)**

1. Connect to the database as a user with DBA or RESOURCE authority.
2. Execute a DROP TEXT INDEX statement.

   For example, to drop the MyTextIndex TEXT index, enter:

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

Refreshing the TEXT index

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

Deleting rows in a TEXT index

The TEXT_DELETE_METHOD database option specifies the algorithm used during a delete in a TEXT index.

TEXT_DELETE_METHOD option

Function	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

DBA permissions are not required to set this option. Can be set temporary, for an individual connection, or for the PUBLIC group. Takes effect immediately.

Description

This option chooses 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
```

See also

See “Optimizing delete operations” in Chapter 3, “Optimizing Queries and Deletions” in the *Performance and Tuning Guide*.

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 a Sybase 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.

Default text configuration objects

Sybase IQ provides the default text configuration object default_char for use with non-NCHAR data. This configuration is created the first time you create a text configuration object or TEXT index. In addition, 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 Sybase IQ tables.

Table 2-2 shows the default settings for default_char and default_nchar, which are best suited for most character-based languages. Sybase strongly recommends that you do not change the settings in the default text configuration objects.

Table 2-2: 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.

Creating a text configuration

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

❖ **Creating a text configuration (Sybase Central)**

1. Connect to the database as a user with DBA or RESOURCE authority.
2. In the left pane, right-click the Text Configurations Objects folder and select New > Text Configuration Object.
3. Type a name for the text configuration.
4. Select the owner of the text configuration.
5. Select the type of database collation for the text configuration. Click Next.

   Note Text configurations with NCHAR collation are not supported by Sybase IQ TEXT indexes.

6. Select the Generic term-breaker algorithm.

   Note Sybase IQ text configurations do not support the N-gram algorithm. See “Text configuration object settings” on page 15.

7. Enter the minimum and maximum term length.
8. If using a external term breaker library, select “Use an external term breaker” and specify the external term breaker function and library.

   Specify the function and library in the form `function-name@library-file-name`.

9. Click Next.
10. If using a external prefilter library, select "Use an external prefilter" and specify the external prefilter function and library.

    Specify the function and library in the form `function-name@library-file-name`.
11. Add any terms to ignore when building a TEXT index with this text configuration to the Stoplist. Separate terms with a space.

    Terms in this list are also ignored in a query.
12. Click Next.
13. Type a comment describing the text configuration, and click Finish.

❖ **Creating a text configuration object (Interactive SQL)**

1. Connect to the database as a user with DBA or RESOURCE authority.
2. Execute a CREATE TEXT CONFIGURATION statement.

   For example, to create a text configuration object called myTxtConfig using the default_char text configuration object as a template, enter:

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

Text configuration object settings

The following tables explain text configuration object settings, how they affect what is indexed, and how a full text search query is interpreted. For examples of text configuration objects and their impact on TEXT indexes and full text searching, see "Text configuration object setting interpretations" on page 21.

Term breaker algorithm (*TERM BREAKER*) The TERM BREAKER setting specifies the algorithm to use for breaking strings into terms. Sybase IQ supports GENERIC (the default) for storing terms.

Note NGRAM term breakers for storing n-grams (an n-gram is a group of characters of length *n* where *n* is the value of MAXIMUM TERM LENGTH) are supported only in IN SYSTEM tables, and cannot be used in Sybase IQ TEXT indexes.

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 2-3: 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.
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.

*NGRAM TEXT indexes are supported only in IN SYSTEM tables.

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 2-4: 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.

*NGRAM TEXT indexes are supported only in IN SYSTEM tables.

Maximum term length setting (*MAXIMUM TERM LENGTH*) 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 it 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 2-5: 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.

*NGRAM TEXT indexes are supported only in IN SYSTEM tables.

Stoplist setting (*STOPLIST*) The stoplist setting specifies terms that are not indexed. The default for this setting is taken from the setting in the default text configuration object, which typically has an empty stoplist.

Table 2-6: 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.

*NGRAM TEXT indexes are supported only in IN SYSTEM tables.

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.

Displaying a list of text configurations

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

❖ **Displaying a list of text configurations (Sybase Central)**

1 Connect to the database as a user with DBA or RESOURCE authority.

2 In the left pane, select Text Configurations Objects.

A list of all text configurations appears in the right pane.

❖ **Displaying a list of text configurations (Interactive SQL)**

1 Connect to the database as a user with DBA or RESOURCE authority.

2 Execute a SELECT statement.

For example, to list all text configuration objects, use:

```
SELECT * FROM SYSTEXTCONFIG;
```

Altering a text configuration

Change the settings of the text configuration object, including the dbspace and permitted term lengths range. You can alter only text configuration objects that are not being used by a TEXT index.

❖ **Altering a text configuration object (Sybase Central)**

1 Connect to the database as a user with DBA or RESOURCE authority.

2 In the left pane, select Text Configurations Objects.

3 In the list of Text Configurations, right-click the object to modify and select Properties.

4 Switch to the Settings tab, and modify the settings as needed.

5 Click OK.

❖ **Altering a text configuration object (Interactive SQL)**

1. Connect to the database as a user with DBA or RESOURCE authority, or as the owner of the text configuration object.
2. Execute an ALTER TEXT CONFIGURATION statement.

   For example, to alter the minimum term length for the myTxtConfig text configuration object, use:

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

Modifying the stoplist

The stoplist contains a list of terms to ignore when building a TEXT index with this text configuration. You can alter only text configuration objects that are not being used by a TEXT index.

❖ **Modifying the stoplist (Sybase Central)**

1. Connect to the database as a user with DBA or RESOURCE authority.
2. In the left pane, select Text Configurations Objects.
3. In the list of Text Configurations, right-click the object to modify and select Properties.
4. Switch to the Stoplist tab, and modify the stoplist words as needed. Use a space to separate the terms.
5. To sort the list of stoplist terms alphabetically and show them in a list, click Sort Terms.
6. When the stoplist is updated, click OK.

❖ **Modifying the stoplist (Interactive SQL)**

1. Connect to the database as a user with DBA or RESOURCE authority.
2. Execute a ALTER TEXT CONFIGURATION statement with the STOPLIST clause.

   For example, to add a stoplist to the myTxtConfig configuration object, use:

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

Dropping a text configuration

Remove an unnecessary text configuration from the database. Only text configurations that are not being used by a TEXT index can be dropped.

❖ **Dropping a text configuration (Sybase Central)**

1. Connect to the database as a user with DBA or RESOURCE authority.
2. In the left pane, select Text Configurations Objects.
3. In the list of Text Configurations, right-click the object to modify and select Delete.
4. In the confirmation dialog, click Yes.

❖ **Dropping a text configuration (Interactive SQL)**

1. Connect to the database as a user with DBA or RESOURCE authority.
2. Execute a DROP TEXT CONFIGURATION statement.

   For example, to drops a text configuration object called myTxtConfig, use:

   ```
   DROP TEXT CONFIGURATION myTxtConfig;
   ```

Text configuration object examples

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

Text configuration object setting interpretations

Table 2-7 shows the settings for different text configuration objects and 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 2-7: 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'`

Text configuration object CONTAINS query string interpretations

Table 2-8 provides examples of how the settings of the text configuration object strings are interpreted.

The parenthetical numbers in the Interpreted string column 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; and symbols like *, I, and others are dropped as they are not alphanumeric.

Table 2-8: CONTAINS string interpretations

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

Limiting prefix terms in a text search

The MAX_PREFIX_PER_CONTAINS_PHRASE database option specifies the number of prefix terms allowed in a text search expression.

MAX_PREFIX_PER_CONTAINS_PHRASE option

Function

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

DBA permissions are not required to set this option. Can be set temporary, for an individual connection, or for the PUBLIC group. 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. Sybase 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, 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.

CHAPTER 3 External Libraries

This chapter describes using external libraries to supply prefiltering and term breaking for documents.

Understanding external libraries

Sybase 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 a Sybase-certified partner. For a information on certified vendor solutions, see the Partner Certification Reports web site at http://www.sybase.com/detail_list?id=9784 and then filter the certification reports to show Sybase 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 Sybase IQ server can explicitly enable or disable this functionality. See "Enabling and disabling external libraries on startup" on page 27.

The ISYSTEXTCONFIG system table stores information about the external libraries associated with a text configuration object. See "SYSTEXTCONFIG system view" in Chapter 8, "System Tables and Views" of *Reference: Building Blocks, Tables, and Procedures*.

External library restrictions

Sybase IQ text configuration objects and TEXT indexes using external libraries have these limitations:

- For TEXT indexes on binary columns, you must use external libraries provided by external vendors for document conversion. Sybase 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.
- You cannot use external libraries to create TEXT indexes on SQL Anywhere tables; doing so results in an error.

Using 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.

Enabling and disabling external libraries on startup

Sybase IQ provides a 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.

Unloading external libraries

Use the system procedure dbo.sa_external_library_unload to unload an external library when the library is not in use. The procedure takes one optional parameter, a long varchar. The parameter specifies the name of the library to be unloaded. If you do not specify a parameter, all external libraries that are not in use are unloaded.

This example unloads an external function library.

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

CHAPTER 4 Unstructured Data Queries

This chapter describes querying large object data, including the full text search capability that handles unstructured and semistructured data.

Full text searching

Full text search can quickly find all instances of a term (word) in a database without having to scan table rows. Full text search uses TEXT indexes, which 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. See “CONTAINS conditions” on page 31 and “CONTAINS conditions” in Chapter 2, “SQL Language Elements” in *Reference: Building Blocks, Tables, and Procedures*.

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.

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.

FROM clause

Description

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.

See “CONTAINS conditions” on page 31 and CONTAINS conditions in Chapter 2, “SQL Language Elements,” in *Reference: Building Blocks, Tables, and Procedures*.

For the full syntax and description of the FROM clause, see FROM clause in Chapter 1, “SQL Statements” of *Reference: Statements and Options*.

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** Use *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** Use *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** Use *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** Use fuzzy-expression to find terms that are similar to what you specify. Fuzzy matching is only supported on NGRAM TEXT indexes for IN SYSTEM tables.
- **proximity-expression** 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 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** Use *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.

Remarks

The CONTAINS search condition takes a column list and *contains-query-string* as arguments. It 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 SQL Anywhere table with a full TEXT index that is joined to a remote table.
- Queries using Sybase IQ and SQL Anywhere tables, where the full TEXT index to be used is on the 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: Interpreting operators and precedence** During this step, keywords are interpreted as operators, and rules of precedence are applied.
- **Step 2: Applying 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.

Operator precedence in a CONTAINS search condition

During query evaluation, expressions are evaluated using the following order of precedence:

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

Allowed syntax for asterisk (*)

The asterisk is used for prefix searching. 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.

Table 4-1 shows allowable asterisk usage:

Table 4-1: 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.

Allowed syntax for hyphen (-)

The hyphen can be used 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.

Table 4-2 shows the allowed syntax for hyphen:

Table 4-2: 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.
`'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.

Query string	Equivalent to	Interpreted as
`'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.

Allowed syntax for special characters

Table 4-3 shows the allowed syntax for all special characters except asterisk and hyphen. These 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 4-3: 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'`
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.

Effect of dropped terms

TEXT indexes are built according to the settings defined for the text configuration object used to create the TEXT index. ATEXT 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, Sybase IQ returns the error `CONTAINS has NULL search term`. SQL Anywhere reports no error and returns zero rows.

CONTAINS table expressions

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;
```

Note The SQL Anywhere documentation provides full text search examples. Not all of these examples apply to Sybase IQ 15.2. For example, Sybase IQ does not support text searches that are part of the IF search condition and does not allow fuzzy or NGRAM searches.

See also

“Types of full text searches” in *SQL Anywhere 11.0.1 > SQL Anywhere Server - SQL Usage > Querying and Modifying Data > Querying data*.

Querying 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.

Sybase 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`.

See Chapter 9, "Function Support."

Querying 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).

See Chapter 9, "Function Support."

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.

See "CONTAINS conditions" in Chapter 2, "SQL Language Elements" in *Reference: Building Blocks, Tables, and Procedures*.

Monitoring performance of LONG BINARY and LONG VARCHAR columns

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

CHAPTER 5 Stored Procedure Support

This chapter describes stored procedure support for the LONG BINARY (BLOB) and LONG VARCHAR (CLOB) data type columns, and full text searching.

Managing terms in a TEXT index

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

sa_char_terms system procedure

Function

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.

Description

You can use this system procedure 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.

Permissions

None.

Examples

This statement returns 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 5-1: CHAR string interpretation

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

sa_nchar_terms system procedure

Function

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.

Description

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.

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

Permissions

None

Examples

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

sa_text_index_stats system procedure

Function

Returns statistical information about the TEXT indexes in the database.

Syntax

sa_text_index_stats( )

Description

Use sa_text_index_stats to view statistical information for each TEXT index in the database. This table describes the information returned by sa_text_index_stats:

Table 5-2: Statistical information for TEXT indexes

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_config_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.

Permissions

DBA authority required

Examples

This statement returns statistical information for each TEXT index in the database:

```
CALL sa_text_index_stats( );
```

sa_text_index_vocab system procedure

Function

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

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.

Description

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

sa_text_index_vocab has this limitation:

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

Permissions

DBA authority, or SELECT permission on the indexed table is required.

Examples

This example executes 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 5-3: Terms in the index

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

Identifying external libraries

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.

sa_external_library_unload system procedure

Function

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.

Description

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.

Permissions

DBA authority required.

Examples

This example unloads an external library called *myextlib.dll*:

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

This example unloads all libraries that are not currently in use:

```
CALL sa_external_library_unload();
```

sa_list_external_library procedure

Function	Lists the external libraries currently loaded in the server.
Syntax	**sa_list_external_library**( )
Description	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.
Permissions	DBA authority required.
Examples	This example lists the external libraries and their reference count:

```
call sa_list_external_library()
```

Controlling large object data compression

The sp_iqsetcompression stored procedure 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.

sp_iqsetcompression procedure

Function	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	Requires DBA authority.
Description	sp_iqsetcompression provides control of compression of LONG BINARY (BLOB) and LONG VARCHAR (CLOB) data type columns. The compression setting applies only to Sybase IQ base tables.

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

Table 5-4: 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

For this 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 (you must have DBA permission):

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

This command returns no rows.

sp_iqshowcompression procedure

Function

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

Syntax

sp_iqshowcompression (*owner*, *table*, *column*)

Permissions

Requires DBA authority.

Description

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

Table 5-5: 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

For this 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 (you must have DBA permission):

```
CALL sp_iqshowcompression('USR', 'pixTable',
'picJPG') ;
```

This command returns one row:

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

Displaying information about large object columns

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

Size of a LONG BINARY column

This output 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%.

Size of a LONG VARCHAR column

This output 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%.

C H A P T E R 6

Large Object Data Load and Unload

This chapter describes how to export and load large object data in Sybase IQ.

Exporting large object data

The 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.

BFILE function

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.

Example

This example shows how to use BFILE to extract data from the LONG BINARY column lobcol, which is created and loaded in the “Load example” on page 52. To write the data in files that can be used as secondary files in a load, enter:

```
SELECT c1, filename, ext,
‘../myoutput/’ + TRIM(filename) + ‘.’ + TRIM(ext) fname,
BFILE(fname, lobcol)
FROM ltab
   WHERE lobcol IS NOT NULL
   AND ext IS NOT NULL
```

This command generates the file name with extension *boston.jpg* for lobcol in row 1 and the file name with extension *map_of_concord.bmp* for lobcol in row 2.

Loading large object data

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.

Extended LOAD TABLE syntax

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.

Sybase 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, Sybase 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.

Sybase 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 details on loading data using binary format, see “Using binary load format” in Chapter 7, “Moving Data In and Out of Databases” of the *System Administration Guide: Volume 1*.

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. See LOAD TABLE statement in Chapter 1, “SQL Statements” in *Reference: Statements and Options*.

Load example

This example shows the SQL statements to 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.

Controlling 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 group or temporary; it 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

Stripping 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.

Enclosing 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.

Truncating 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.

Table 6-1 lists how a trailing multibyte character is loaded, depending on the values of TRIM_PARTIAL_MBC and SECONDARY_FILE_ERROR.

Table 6-1: 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

See also

For information on support of large object variables by the LOAD TABLE, INSERT…VALUES, INSERT…SELECT, INSERT…LOCATION, SELECT…INTO, and UPDATE SQL statements, see Chapter 7, “Large Object Data Types.”

CHAPTER 7 Large Object Data Types

This chapter describes the characteristics of the large object LONG BINARY and LONG VARCHAR data type columns and index support of large object data.

Large object data types LONG BINARY and BLOB

Binary large object (BLOB) data in Sybase 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 or join 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 Adaptive Server Enterprise IMAGE column into a LONG BINARY column using the INSERT...LOCATION command. All IMAGE data inserted is silently right-truncated at 2147483648 bytes (2GB).

Data type conversion

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. See Chapter 9, “Function Support.”

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.

Large object data types LONG VARCHAR and CLOB

Character large object (CLOB) data in Sybase 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.

Sybase 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 Adaptive Server Enterprise TEXT column into a LONG VARCHAR column using the INSERT...LOCATION command. All TEXT data inserted is silently right-truncated at 2147483648 bytes (2GB).

Data type conversion

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. See Chapter 9, "Function Support."

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.

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.

Large object variable data type conversion

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

ENABLE_LOB_VARIABLES option

Function	Controls the data type conversion of large object variables.
Allowed values	ON, OFF
Default	OFF
Scope	DBA permissions are not required to set this option. Can be set temporary, for an individual connection, or for the PUBLIC group. Takes effect immediately.
Description	This option 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	To retain the data type and size of large object variables greater than 32K:

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

Index support of large object columns

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

TEXT index support of large object columns

The Sybase IQ TEXT index supports LONG BINARY and LONG VARCHAR columns.

See Chapter 8, "SQL Statement Support" and Chapter 2, "TEXT Indexes and Text Configuration Objects."

WD index support of LONG VARCHAR (CLOB) columns

Sybase IQ offers this support for the WORD (WD) index on LONG VARCHAR (CLOB) columns:

- You can create a WD index on columns of CHAR, VARCHAR, and LONG VARCHAR data types in Sybase Central.
- 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 Sybase 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.

C H A P T E R 8

SQL Statement Support

This chapter describes the SQL statements and syntax that support working with TEXT indexes and text configurations.

ALTER TEXT CONFIGURATION statement

Description

Alters a text configuration object.

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*

stoplist : *string-expression*

library-and-entry-point-name-string : *[operating-system:]function-name@library*

Examples

Example 1 These statements create 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 This statement adds stoplist terms to the maxTerm16 configuration object:

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

Example 3 This example 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 This example 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

TEXT indexes are dependent on a text configuration object. Sybase 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

STOPLIST clause Use this clause 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. Stoplist terms 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 clause Use this clause to drop the stoplist for a text configuration object.

MINIMUM TERM LENGTH clause 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 clause With GENERIC TEXT indexes, 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 clause 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. Only the GENERIC term breaker is supported by Sybase IQ TEXT indexes. 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.

TERM BREAKER can include the specification for the external term breaker library using EXTERNAL NAME and the library entry point.

DROP PREFILTER clause Drops the external prefilter and sets NULL to the prefilter columns in ISYSTEXTCONFIG table.

PREFILTER EXTERNAL NAME clause Specifies the entry_point and the library name of the external pre-filter library provided by external vendors.

Side effects

Automatic commit.

Permissions

The user must have DBA authority to alter the text configuration object to specify the external libraries and functions for external pre-filter or term-breaker.

All other modifications to the text configuration can be done by either the owner of the configuration object or by a user having DBA authority.

ALTER TEXT INDEX statement

Description

Alters the definition of a TEXT index.

Syntax

ALTER TEXT INDEX [*owner.***]text-index-name   ON [** *owner.***]***table-name*
  alter-clause

alter-clause :
  rename-object
  | move-object

rename-object :
  RENAME { AS | TO } *new-name*

move-object:
  MOVE TO *dbspace-name*

Examples

In this example, the first statement creates a TEXT index, MyTextIndex, defining it as IMMEDIATE REFRESH. The second statement renames the TEXT index to Text_index_daily. The third statement moves it 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

RENAME clause   Rename the TEXT index.

MOVE clause   Moves the TEXT index to the specified dbspace.

Side effects

Automatic commit.

Permissions

Must be the owner of the underlying table, or have DBA authority, or have REFERENCES permission to rename the index.

To move the TEXT index, you must have DBA or SPACE ADMIN authority, or you must be the table owner and have CREATE permission on the dbspace.

CREATE TEXT CONFIGURATION statement

Description

Creates a text configuration object.

Syntax

CREATE TEXT CONFIGURATION [*owner*.]*new-config-name*
  FROM [*owner*.]*existing-config-name*

Examples

This CREATE TEXT CONFIGURATION statement creates a text configuration object, max_term_sixteen, using the default_char text configuration object. The subsequent ALTER TEXT CONFIGURATION statement changes 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

Create a text configuration object using another text configuration object as a template and 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

FROM clause   Specify the name of a text configuration object to use as the template for creating the new one. The names of the default text configuration objects are default_char and default_nchar. Only default_char is supported for Sybase IQ tables; default_nchar is supported only on SQL Anywhere tables.

Side effects

Automatic commit.

Permissions

Must have DBA or RESOURCE authority.

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

CREATE TEXT INDEX statement

Description

Creates a TEXT index.

Syntax

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

Examples

This example creates a TEXT index, myTxtIdx, on the CompanyName column of the Customers table in the iqdemo database. The max_term_sixteen text configuration object is used.

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

Usage

You cannot create a TEXT index on views or temporary tables. You cannot create a TEXT index on an IN SYSTEM materialized view.

TEXT indexes will not be replicated to join indexes tables. A TEXT index can be created on a column of a table that participates in a join index.

The BEGIN PARALLEL IQ…END PARALLEL IQ statement does not support CREATE TEXT INDEX.

ON Clause Specifies the table and column on which to build the TEXT index.

IN clause 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 clause 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.

REFRESH clause IMMEDIATE REFRESH is used as the default and is the only permitted value for tables in Sybase IQ. Specify IMMEDIATE REFRESH to refresh the TEXT index each time changes in the underlying table impact data in the TEXT index.

An IMMEDIATE REFRESH TEXT index is populated at creation and is refreshed whenever the data in the underlying column is changed. Once a TEXT index is created, you cannot change it to, or from, IMMEDIATE REFRESH.

Side effect

Automatic commit.

Permissions

You must be the owner of the underlying table, or have DBA authority, or have REFERENCES permission.

You must have CREATE permission on the dbspace.

DROP TEXT CONFIGURATION statement

Description

Drops a text configuration object.

Syntax

DROP TEXT CONFIGURATION [*owner*.]*text-config-name*

Examples

These statements create and drop the mytextconfig text configuration object:

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

Usage

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

Must be the owner of the text configuration object or have DBA authority.

DROP TEXT INDEX statement

Description

Removes a TEXT index from the database.

Syntax

DROP TEXT INDEX*text-index-name*
  ON[*owner*] *table-name*

Examples

These statements create and drop the TextIdx TEXT index:

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

Usage

ON specifies the table on which the TEXT index was built.

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

Side effects

Automatic commit

Permissions

Must be the owner of the underlying table, or have DBA authority, or have REFERENCES permission.

CHAPTER 9 Function Support

This chapter describes the Sybase IQ functions that support the LONG BINARY and LONG VARCHAR data types.

Summary of function support of large object data

The Table 9-1 summarizes the function support of LONG BINARY (BLOB) and LONG VARCHAR (CLOB) data types and LONG BINARY and LONG VARCHAR variables.

In addition to the functions described in this chapter, you can use the BFILE function to extract LOB data. See "Exporting large object data" on page 49.

Table 9-1: 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.

The following sections describe the functions that support LONG BINARY and LONG VARCHAR columns and variables. For full descriptions of these functions and examples, see Chapter 4, “SQL Functions” of *Reference: Building Blocks, Tables, and Procedures*.

BIT_LENGTH function

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 Sybase 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.

BYTE_LENGTH function

Function

The BYTE_LENGTH function returns the number of bytes in a string.

Usage

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.

For BYTE_LENGTH function syntax and usage, see “BYTE_LENGTH function [String]” in Chapter 4, “SQL Functions” in *Reference: Building Blocks, Tables, and Procedures*.

BYTE_LENGTH64 function

Function

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.

BYTE_SUBSTR64 and BYTE_SUBSTR functions

Function

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.

CHAR_LENGTH function

Function

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.

CHAR_LENGTH64 function

Function

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.

CHARINDEX function

Function

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.

See also

See “CHARINDEX function [String]” in Chapter 4, “SQL Functions” in *Reference: Building Blocks, Tables, and Procedures*.

LOCATE function

Function

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.

See also

"LOCATE function [String]" in Chapter 4, "SQL Functions" in *Reference: Building Blocks, Tables, and Procedures*

OCTET_LENGTH function

Function

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 Sybase 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.

PATINDEX function

Function

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.

See also

- "PATINDEX function [String]" in Chapter 4, "SQL Functions" in *Reference: Building Blocks, Tables, and Procedures*
- "LIKE conditions" in Chapter 2, "SQL Language Elements" in *Reference: Building Blocks, Tables, and Procedures*

SUBSTRING function

Function

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.

SUBSTRING64 function

Function

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

Given a column named col1 that contains the string (“ABCDEFG”), the SUBSTRING64 function returns the following values:

`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.

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.

APPENDIX A Error and Warning Messages

About this appendix

This appendix describes the error and warning messages that may be returned when you are working with unstructured data, including LONG BINARY and LONG VARCHAR columns.

Contents

Error 1000195

Message text

LOAD specification ‘%2’ only valid for column(s) having datatype ‘%3’. %1

Item	Value
SQLCode	-1000195L
Constant	EMSG_BINARYFILE
SQLState	QDB95
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	20855
Severity Code	14
Parameter 1	location of the exception
Parameter 2	type of load specification
Parameter 3	data type of column

Probable cause

The named load specification in a LOAD TABLE statement is only valid for columns with the given data type.

Error 1000198

Message text

Cannot create join index with table(s) having column(s) of datatype %2. %1

Item	Value
SQLCode	-1000198L
Constant	EMSG_CANNOT_CREATE_JOIN_INDEX
SQLState	QDB98
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	20858
Severity Code	14
Parameter 1	location of the exception
Parameter 2	data type of column

Probable cause

This error is reported when you attempt to create a join index on a table that has one or more LONG VARCHAR or LONG BINARY data type columns. The JOIN INDEX functionality is supported for most data types. There are a few data types, however, for which this functionality is not supported (for example, LONG BINARY and LONG VARCHAR).

Error 1000332

Message text

Odd length of binary data value detected on column %2 %1

Item	**Value**
SQLCode	-1000332L
Constant	EMSG_ODDNUMBER_NIBBLES
SQLState	QDD20
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	21206
Severity Code	14
Parameter 1	location of the exception
Parameter 2	column name

Probable cause

When binary data of hexadecimal format is loaded into a LONG BINARY column from a primary load file, Sybase IQ requires that the total number of hexadecimal digits is an even number. This error 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.

Error 1001013

Message text

Invalid data type comparison %1

Item	Value
SQLCode	-1001013L
Constant	EMSG_TYPECOMPAREERROR
SQLState	QFA13
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	20522
Severity Code	14
Parameter 1	location of the exception

Probable cause

This error is reported if you attempt to search for a pattern in a LONG BINARY column using a LIKE predicate. LIKE predicates are not supported on LONG BINARY (BLOB) columns.

Error 1001051

Message text

Query returns %3 data > 2GB. Use %2 %1

Item	Value
SQLCode	-1001051L
Constant	EMSG_LOB_OVER_2G_W_ARG
SQLState	QFA47
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	21097
Severity Code	14
Parameter 1	SA parse source code line
Parameter 2	function recommended
Parameter 3	long binary or long varchar data type

Probable cause

This error is reported when a query attempts to return a LONG BINARY or LONG VARCHAR value greater than 2 gigabytes.

Error 1001052

Message text

Parameter %2 must be long binary/varchar type. %3 %1

Item	Value
SQLCode	-1001052L
Constant	EMSG_ONLY_SUPPORT_LOB_W_ARG
SQLState	QFA48
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	21098
Severity Code	14
Parameter 1	SA parse source code line
Parameter 2	LOB argument name
Parameter 3	recommended function name

Probable cause

This error is reported when an invalid data type is used for a Large Object (LOB) function parameter.

Error 1001053

Message text

Wrong number of parameters to function %2 %1

Item	Value
SQLCode	-1001053L
Constant	EMSG_WRONG_NUM_PARAMS_W_ARG
SQLState	QFA49
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	21099
Severity Code	14
Parameter 1	SA parse source code line
Parameter 2	function name

Probable cause

This error is reported when a Large Object (LOB) function is passed an incorrect number of arguments.

Error 1001054

Message text

You cannot specify long binary/varchar column in the ORDER/GROUP by clause or in an aggregate function. %1

Item	Value
SQLCode	-1001054L
Constant	EMSG_LOB_NOT_ALLOWED_GROUP
SQLState	QFA50
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	21100
Severity Code	14
Parameter 1	location of the exception

Probable cause

This error is reported when you attempt to use a LONG BINARY column in an ORDER BY, GROUP BY, or aggregation clause.

Warning 1001055

Message text

An error occurred loading %1 column, %2, for %3, rowid %4.

Item	Value
SQLCode	1001055L
Constant	EMSG_LOB_LOAD_ERROR_WARN
SQLState	QFA51
ODBC 2 State	OK
ODBC 3 State	OK
Sybase Error Code	21101
Severity Code	10
Parameter 1	long binary or long varchar data type
Parameter 2	FP index name
Parameter 3	secondary file name
Parameter 4	rowid

Probable cause

This warning message is returned when an error is encountered either opening or reading a LONG BINARY or LONG VARCHAR secondary file during a load operation. This warning message is returned in the server log and the IQ message file when the SECONDARY_FILE_ERROR option is OFF and an error occurs.

Warning 1001056

Message text

An error occurred extracting %1 column, %2, for %3.

Item	Value
SQLCode	1001056L
Constant	EMSG_LOB_EXTRACT_ERROR_WARN
SQLState	QFA52
ODBC 2 State	OK
ODBC 3 State	OK
Sybase Error Code	21102
Severity Code	10
Parameter 1	long binary or long varchar data type
Parameter 2	FP index name
Parameter 3	secondary file name

Probable cause

This warning message is returned when you attempt to extract a LONG BINARY or LONG VARCHAR column and an error is encountered during the extract operation. This warning message is returned in the server log and the IQ message file when the SECONDARY_FILE_ERROR option is OFF and an error occurs.

Error 1001057

Message text

You must use BFILE() to extract %2 column. %1

Item	Value
SQLCode	-1001057L
Constant	EMSG_LOB_EXTRACT_USE_BFILE
SQLState	QFA53
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	21103
Severity Code	14
Parameter 1	location of the exception
Parameter 2	long binary or long varchar data type

Probable cause

This error is reported when you execute a query containing a LONG BINARY or LONG VARCHAR column with the database option TEMP_EXTRACT_NAME1 set ON and you did not specify the BFILE function.

Error 1001058

Message text

The secondary file name, %2, is too long. %1

Item	Value
SQLCode	-1001058L
Constant	EMSG_LOB_SECONDARY_FILE_TOOLONG
SQLState	QFA54
ODBC 2 State	OK
ODBC 3 State	OK
Sybase Error Code	21104
Severity Code	14
Parameter 1	location of the exception
Parameter 2	secondary file name

Probable cause

This error is reported when the length of the LOAD TABLE secondary file pathname exceeds the pathname length limit of the operating system. The action taken when this error is reported depends on the value of the SECONDARY_FILE_ERROR database option.

Error 1009189

Message text

Text document exceeds maximum number of terms. Support up to 4294967295 terms per document. %1

Item	**Value**
SQLCode	-1009189L
Constant	EMSG_MAXTERM_ERROR
SQLState	QSB84
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	21210
Severity Code	14
Parameter 1	location of the exception

Probable cause

Error from external prefilter or term breaker library.

Error 1012030

Message text

for long binary/varchar Column ‘%2’, database page size of (%3) must be greater than %4. %1

Item	Value
SQLCode	-1012030
Constant	EMSG_CAT_PAGESIZETOOSMALL
SQLState	QUA30
ODBC 2 State	ERROR
ODBC 3 State	ERROR
Sybase Error Code	20953
Severity Code	14
Parameter 1	location of the exception
Parameter 2	column number
Parameter 3	requested page size
Parameter 4	minimum allowed page size

Probable cause

The database page size is too small to create a LONG BINARY or LONG VARCHAR column. The database page size must be 128K or greater to create a LONG BINARY or LONG VARCHAR column.

Index