Large Objects Management in Sybase IQ

Sybase® IQ
12.7
### Contents

About This Book ............................................................................................................................ v

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction to Large Objects Management in Sybase IQ</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>The Large Objects Management option</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Binary Large Object (BLOB) data</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Large Object data types LONG BINARY and BLOB</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>LONG BINARY columns in queries</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Host variables of LONG BINARY data type</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Monitoring performance of LONG BINARY columns</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Character Large Object (CLOB) data</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Large Object data types LONG VARCHAR and CLOB</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>LONG VARCHAR columns in queries</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Host variables of LONG VARCHAR data type</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Monitoring performance of LONG VARCHAR columns</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Function Support</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Function support of LONG BINARY data type</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Function support of LONG VARCHAR data type</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Stored Procedure Support</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Controlling large object data compression</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td><code>sp_iqsetcompression</code> procedure</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td><code>sp_iqshowcompression</code> procedure</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Displaying information about large object columns</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>Moving Large Object Data</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Exporting large object data</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Loading large object data</td>
<td>22</td>
</tr>
</tbody>
</table>
Large Objects Management

About This Book

Sybase IQ is a high-performance decision support server designed specifically for data warehouses and data marts. This book, *Large Objects Management in Sybase IQ*, provides reference material for working with Large Object (LOB) data in Sybase IQ. This manual is the place to look for information such as available syntax, parameters, functions, stored procedures, and options related to Sybase IQ LOB data. Read this manual to understand storage and retrieval of Binary Large Objects (BLOBs) and Character Large Objects (CLOBs) within the Sybase IQ data repository.

This manual is a reference for all users of Sybase IQ.

This book provides descriptions of the Large Objects Management features in Sybase IQ and is designed to be used as a reference together with the other books in the Sybase IQ documentation set.

The following table shows which chapters fit a particular interest or need.

**Table 1: Guide to using this book**

<table>
<thead>
<tr>
<th>To learn about...</th>
<th>Refer to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>The characteristics of Binary Large Object data</td>
<td>Chapter 2, “Binary Large Object (BLOB) data”</td>
</tr>
<tr>
<td>The characteristics of Character Large Object data</td>
<td>Chapter 3, “Character Large Object (CLOB) data”</td>
</tr>
<tr>
<td>Functions that support Large Object data</td>
<td>Chapter 4, “Function Support”</td>
</tr>
<tr>
<td>Stored procedures that support Large Object data</td>
<td>Chapter 5, “Stored Procedure Support”</td>
</tr>
<tr>
<td>Exporting and loading Large Object data</td>
<td>Chapter 6, “Moving Large Object Data”</td>
</tr>
<tr>
<td>Compatibility of Sybase IQ Large Object data</td>
<td>Chapter 7, “Compatibility and Conformance”</td>
</tr>
<tr>
<td>Errors and warnings specific to Large Object data</td>
<td>Chapter 8, “Error and Warning Messages”</td>
</tr>
<tr>
<td>Upgrading Sybase IQ 12.5 LONG BINARY columns</td>
<td>Appendix A, “Upgrading existing LONG BINARY columns”</td>
</tr>
</tbody>
</table>

**Windows platforms**

The Windows information in this book applies to all supported Windows platforms, unless noted otherwise. For supported Windows platforms, see the *Release Bulletin Sybase IQ for Windows*.
Documentation for Sybase IQ:

- **Introduction to Sybase IQ**
  Read and try the hands-on exercises if you are unfamiliar with Sybase IQ or with the Sybase Central™ database management tool.

- **New Features in Sybase IQ 12.7**
  Read just before or after purchasing Sybase IQ for a list of new features.

- **Sybase IQ Performance and Tuning Guide**
  Read to understand query optimization, design, and tuning issues for very large databases.

- **Sybase IQ Reference Manual**
  Read for a full description of the SQL language, stored procedures, data types, and system tables supported by Sybase IQ.

- **Sybase IQ System Administration Guide**
  Read to understand administration issues such database creation and load operations, data security and integrity, server startup and connection, and multiplex operations.

- **Sybase IQ Troubleshooting and Recovery Guide**
  Read to solve problems, perform system recovery, and repair databases.

- **Sybase IQ Error Messages**
  Refer to Sybase IQ error messages (referenced by SQLCode, SQLState, and Sybase error code) and SQL preprocessor errors and warnings.

- **Sybase IQ Utility Guide**
  Read for Sybase IQ utility program reference material, such as available syntax, parameters, and options.

- **Sybase IQ Installation and Configuration Guide**
  Read the edition for your platform before and while installing Sybase IQ, when migrating to a new version of Sybase IQ, or when configuring Sybase IQ for a particular platform.

- **Sybase IQ Release Bulletin**
  Read just before or after purchasing Sybase IQ for last minute changes to the product and documentation. Read for help if you encounter a problem.

- **Encrypted Columns in Sybase IQ**
Read to understand the use of user encrypted columns within the Sybase IQ data repository. You need a separate license to install this product option.

Sybase IQ and Adaptive Server Anywhere
Because Sybase IQ is an extension of Adaptive Server® Anywhere, a component of SQL Anywhere® Studio, IQ supports many of the same features as Adaptive Server Anywhere. The IQ documentation set refers you to SQL Anywhere Studio documentation where appropriate.

Documentation for Adaptive Server Anywhere:

- **Adaptive Server Anywhere Programming Guide**
  Intended for application developers writing programs that directly access the ODBC, Embedded SQL™, or Open Client™ interfaces, this book describes how to develop applications for Adaptive Server Anywhere.

- **Adaptive Server Anywhere Database Administration Guide**
  Intended for all users, this book covers material related to running, managing, and configuring databases and database servers.

- **Adaptive Server Anywhere SQL Reference Manual**
  Intended for all users, this book provides a complete reference for the SQL language used by Adaptive Server Anywhere. It also describes the Adaptive Server Anywhere system tables and procedures.

You can also refer to the Adaptive Server Anywhere documentation in the SQL Anywhere Studio 9.0.2 collection on the Sybase Product Manuals Web site. To access this site, go to Product Manuals at [http://www.sybase.com/support/manuals/](http://www.sybase.com/support/manuals/).

Other sources of information

Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.

- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.
Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at [http://www.sybase.com/support/manuals/](http://www.sybase.com/support/manuals/).

- Infocenter is an online version of SyBooks that you can view using a standard Web browser. To access the Infocenter Web site, go to Sybooks Online Help at [http://infocenter.sybase.com/help/index.jsp](http://infocenter.sybase.com/help/index.jsp).

**Sybase certifications on the Web**

Technical documentation at the Sybase Web site is updated frequently.

- **Finding the latest information on product certifications**
  2. Click Certification Report.
  3. In the Certification Report filter select a product, platform, and timeframe and then click Go.
  4. Click a Certification Report title to display the report.

- **Finding the latest information on component certifications**
  2. Either select the product family and product under Search by Product; or select the platform and product under Search by Platform.
  3. Select Search to display the availability and certification report for the selection.
Creating a personalized view of the Sybase Web site (including support pages)

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

2. Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

Finding the latest information on EBFs and software maintenance

2. Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
3. Select a product.
4. Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

   Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.
5. Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Syntax conventions

This documentation uses the following syntax conventions in syntax descriptions:

- **Keywords**   SQL keywords are shown in UPPER CASE. However, SQL keywords are case insensitive, so you can enter keywords in any case you wish; SELECT is the same as Select which is the same as select.

- **Placeholders**   Items that must be replaced with appropriate identifiers or expressions are shown in *italics*.

- **Continuation**   Lines beginning with ... are a continuation of the statements from the previous line.
• **Repeating items** Lists of repeating items are shown with an element of the list followed by an ellipsis (three dots). One or more list elements are allowed. If more than one is specified, they must be separated by commas.

• **Optional portions** Optional portions of a statement are enclosed by square brackets. For example:

```sql
RELEASE SAVEPOINT [ savepoint-name ]
```

It indicates that the `savepoint-name` is optional. The square brackets should not be typed.

• **Options** When none or only one of a list of items must be chosen, the items are separated by vertical bars and the list enclosed in square brackets. For example:

```sql
[ ASC | DESC ]
```

It indicates that you can choose one of ASC, DESC, or neither. The square brackets should not be typed.

• **Alternatives** When precisely one of the options must be chosen, the alternatives are enclosed in curly braces. For example:

```sql
QUOTES { ON | OFF }
```

It indicates that exactly one of ON or OFF must be provided. The braces should not be typed.

Table 2 lists the typographic conventions used in this documentation.

<table>
<thead>
<tr>
<th><strong>Table 2: Typographic conventions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item</strong></td>
</tr>
<tr>
<td>Code</td>
</tr>
<tr>
<td>User entry</td>
</tr>
<tr>
<td>emphasis</td>
</tr>
<tr>
<td>file names</td>
</tr>
<tr>
<td>database objects</td>
</tr>
</tbody>
</table>
The sample database

Sybase IQ includes a sample database used by many of the examples in the IQ documentation.

The sample database represents a small company. It contains internal information about the company (employees, departments, and financial data), as well as product information (products), sales information (sales orders, customers, and contacts), and financial information (fin_code, fin_data).

The sample database is held in a file named asiqdemo.db, located in the directory $ASDIR/demo on UNIX systems and %ASDIR%\demo on Windows systems.

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Sybase IQ 12.7 and the HTML documentation have been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

For information about accessibility support in the Sybase IQ plug-in for Sybase Central, see “Using accessibility features” in Introduction to Sybase IQ. The online help for this product, which you can navigate using a screen reader, also describes accessibility features, including Sybase Central keyboard shortcuts.

Configuring your accessibility tool

You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool and see “Using screen readers” in Introduction to Sybase IQ.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

For a Section 508 compliance statement for Sybase IQ, go to Sybase Accessibility at http://www.sybase.com/products/accessibility.
If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.
CHAPTER 1

Introduction to Large Objects Management in Sybase IQ

About this chapter
This chapter introduces you to large objects management.

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Large Objects Management option</td>
<td>1</td>
</tr>
</tbody>
</table>

The Large Objects Management option

The Large Objects Management Option extends the capabilities of Sybase IQ Enterprise Edition to allow storage and retrieval of Binary Large Objects (BLOBs) and Character Large Objects (CLOBs) within the Sybase IQ data repository.

*Users must be specifically licensed to use the Large Objects Management functionality described in this product documentation.*

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

- unstructured, in which case the database simply stores and retrieves the data
- structured (for example, text) in which case the database understands the data structure and provides supporting functions (for example, string functions)

Typical LOB data sources are images, maps, documents (for example, PDF), audio, video, and XML. Sizes of an individual LOB may extend into gigabytes (GB), terabytes (TB), or even petabytes (PB).
The Large Objects Management option
CHAPTER 2

Large Objects Management

CHAPTER 2

Binary Large Object (BLOB) data

About this chapter

This chapter describes the characteristics of the LONG BINARY data type column, which stores Sybase IQ Binary Large Object data.

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large Object data types LONG BINARY and BLOB</td>
<td>3</td>
</tr>
<tr>
<td>LONG BINARY columns in queries</td>
<td>5</td>
</tr>
<tr>
<td>Host variables of LONG BINARY data type</td>
<td>5</td>
</tr>
<tr>
<td>Monitoring performance of LONG BINARY columns</td>
<td>5</td>
</tr>
</tbody>
</table>

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.

**Note** Existing LONG BINARY columns created using any Sybase IQ release prior to Sybase IQ 12.5 ESD8 are not supported. All existing LONG BINARY columns created prior to Sybase IQ 12.5 ESD8 must be explicitly dropped before installing Sybase IQ 12.6 or later versions, then recreated after installing Sybase IQ 12.6 or later versions. The ALTER DATABASE UPGRADE command does not upgrade LONG BINARY columns created prior to Sybase IQ 12.5 ESD8. For details on upgrading LONG BINARY columns, see the Appendix, “Upgrading existing LONG BINARY columns.”

LONG BINARY columns created using Sybase IQ 12.5 ESD8 or a later version do not require a special upgrade procedure. Keep in mind, however, that if the server is not licensed for the LOB component, the CREATE TABLE and ALTER TABLE ADD column commands with a LONG BINARY column are not allowed and return the error “Large Objects Management functionality is not licensed on this server.”
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.) The IQ database must be created with an IQ page size of at least 128KB (131072 bytes) in order to accommodate a table with LONG BINARY data.

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.

A non-FP index or join index cannot be constructed on a LONG BINARY column.

A LONG BINARY column can be modified 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.

An Adaptive Server Enterprise IMAGE column can be inserted into a LONG BINARY column using the INSERT...LOCATION command. All IMAGE data inserted is right truncated at 32767 bytes.

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.

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 the section “Function support of LONG BINARY data type” on page 11 for more information on functions that support LONG BINARY.

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, then 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 a rollback.

**LONG BINARY columns in queries**

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

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

See “Function support of LONG BINARY data type” on page 11 for more information on LONG BINARY data and functions.

**Host variables of LONG BINARY data type**

An inbound LONG BINARY host variable (a host variable used by IQ) is limited to a length of 32767 (32K-1) bytes and is handled by IQ as VARBINARY data. An error is raised, if the inbound host variable length is greater than 32767 bytes.

An outbound LONG BINARY host variable (a host variable set by IQ) has a maximum length of 2GB.

**Monitoring performance of LONG BINARY columns**

The Sybase IQ performance monitor displays performance data for LONG BINARY columns.
Monitoring performance of LONG BINARY columns
CHAPTER 3  Character Large Object (CLOB) data

About this chapter
This chapter describes the characteristics of the LONG VARCHAR data type column, which stores Sybase IQ Character Large Object data.

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large Object data types LONG VARCHAR and CLOB</td>
<td>7</td>
</tr>
<tr>
<td>LONG VARCHAR columns in queries</td>
<td>9</td>
</tr>
<tr>
<td>Host variables of LONG VARCHAR data type</td>
<td>9</td>
</tr>
<tr>
<td>Monitoring performance of LONG VARCHAR columns</td>
<td>9</td>
</tr>
</tbody>
</table>

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.) The IQ database must be created with an IQ page size of at least 128KB (131072 bytes) in order to accommodate a table with LONG VARCHAR data.

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 multi-byte 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:

```sql
CREATE TABLE lvtab (c1 INTEGER, c2 CLOB,
                    c3 CLOB NOT NULL);
ALTER TABLE lvtab ADD c4 CLOB;
```

A non-FP index or join index cannot be constructed on a LONG VARCHAR column.

A LONG VARCHAR column can be modified 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.

An Adaptive Server Enterprise TEXT column can be inserted into a LONG VARCHAR column using the INSERT...LOCATION command. All TEXT data inserted is right truncated at 32767 bytes.

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, TENANT, SMALLINT, INT, UNSIGNED INT, BIGINT, UNSIGNED BIGINT, BINARY, VARBINARY, or LONG BINARY data types to LONG VARCHAR data type.

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 the section “Function support of LONG VARCHAR data type” on page 12 for more information on functions that support LONG VARCHAR.

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 a rollback is performed. 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 a rollback is performed. Trailing partial multibyte characters are truncated on conversion.

LONG VARCHAR columns in queries

In WHERE clauses of the SELECT statement, LONG VARCHAR columns can only be used in IS NULL and IS NOT NULL expressions, in addition to the BIT_LENGTH, OCTET_LENGTH, CHAR_LENGTH, CHAR_LENGTH64, SUBSTRING64, and SUBSTRING functions.

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

See “Function support of LONG VARCHAR data type” on page 12 for more information on LONG VARCHAR data and functions.

Host variables of LONG VARCHAR data type

An inbound LONG VARCHAR host variable (a host variable used by IQ) is limited to a length of 32767 (32K-1) bytes. An error is raised, if the inbound host variable length is greater than 32767 bytes.

An outbound LONG VARCHAR host variable (a host variable set by IQ) has a maximum length of 2GB.

Monitoring performance of LONG VARCHAR columns

The Sybase IQ performance monitor displays performance data for LONG VARCHAR columns.
Monitoring performance of LONG VARCHAR columns
This chapter describes the Sybase IQ functions that support the LONG BINARY and LONG VARCHAR data types.

In addition to the functions described in this chapter, the BFILE function can be used to extract LOB data. See “Exporting large object data” on page 21 for more information on the BFILE function.

### Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function support of LONG BINARY data type</td>
<td>11</td>
</tr>
<tr>
<td>Function support of LONG VARCHAR data type</td>
<td>12</td>
</tr>
</tbody>
</table>

### Function support of LONG BINARY data type

The functions BYTE_LENGTH64, BYTE_SUBSTR64, and BYTE_SUBSTR support LONG BINARY data.

The LONG VARCHAR functions BIT_LENGTH, OCTET_LENGTH, and SUBSTRING64 also support LONG BINARY data.

#### BYTE_LENGTH64 function

The BYTE_LENGTH64 function returns an unsigned 64 bit value containing the byte length of the LONG BINARY column parameter.

Syntax:

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

Parameter:

- **large-object-column**  The name of a LONG BINARY column.

#### BYTE_SUBSTR64 and BYTE_SUBSTR functions

The BYTE_SUBSTR64 and BYTE_SUBSTR functions return the long binary byte substring of the LONG BINARY column parameter.

Syntax:

```
BYTE_SUBSTR64( large-object-column, start, length )
```
Function support of LONG VARCHAR data type

BYTE_SUBSTR (large-object-column, start, length)

Parameters:

large-object-column  The name of a LONG BINARY column.

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.

The BYTE_SUBSTR64 and BYTE_SUBSTR functions also support the LONG VARCHAR data type.

Nesting of the functions BYTE_LENGTH64, BYTE_SUBSTR64, and BYTE_SUBSTR is not supported.

Aggregate functions

Only the aggregate function COUNT (*) is supported for LONG BINARY columns. The COUNT DISTINCT parameter is not supported. An error is returned if a LONG BINARY column is used with the MIN, MAX, AVG, or SUM aggregate functions.

Function support of LONG VARCHAR data type

The functions BIT_LENGTH, OCTET_LENGTH, CHAR_LENGTH64, SUBSTRING64, CHAR_LENGTH and SUBSTRING support LONG VARCHAR data.

The LONG BINARY functions BYTE_LENGTH64, BYTE_SUBSTR64, and BYTE_SUBSTR also support LONG VARCHAR data.

See “CHAR_LENGTH function [String]” and “SUBSTRING function [String]” in Chapter 5, “SQL Functions” of the Sybase IQ Reference Manual for descriptions of the CHAR_LENGTH and SUBSTRING functions.

BIT_LENGTH function

The BIT_LENGTH function returns an unsigned 64 bit value containing the bit length of the LONG VARCHAR column parameter. If the argument is NULL, BIT_LENGTH returns NULL.

Syntax:
### BIT_LENGTH(
\textit{column-name})

**Parameter:**

- **\textit{column-name}** The name of a LONG VARCHAR column.

The BIT_LENGTH function supports all Sybase IQ data types.

The OCTET_LENGTH function returns an unsigned 64 bit value containing the byte length of the LONG VARCHAR column parameter. If the argument is NULL, OCTET_LENGTH returns NULL.

**Syntax:**

\begin{verbatim}
OCTET_LENGTH( \textit{column-name} )
\end{verbatim}

**Parameter:**

- **\textit{column-name}** The name of a LONG VARCHAR column.

The OCTET_LENGTH function supports all Sybase IQ data types.

The CHAR_LENGTH function returns a signed 32 bit value containing the character length of the LONG VARCHAR column parameter, including the trailing blanks. If the argument is NULL, CHAR_LENGTH returns NULL. If the character length exceeds 2147483647, an error is returned.

**Syntax:**

\begin{verbatim}
CHAR_LENGTH( \textit{long-varchar-column} )
\end{verbatim}

**Parameter:**

- **\textit{long-varchar-column}** The name of a LONG VARCHAR column.

The CHAR_LENGTH64 function returns an unsigned 64 bit value containing the character length of the LONG VARCHAR column parameter, including the trailing blanks. If the argument is NULL, CHAR_LENGTH64 returns NULL.

**Syntax:**

\begin{verbatim}
CHAR_LENGTH64( \textit{long-varchar-column} )
\end{verbatim}

**Parameter:**

- **\textit{long-varchar-column}** The name of a LONG VARCHAR column.

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

**Syntax:**

\begin{verbatim}
SUBSTRING( \textit{long-varchar-column}, \textit{start [ , \textit{length} ]} )
\end{verbatim}
Function support of LONG VARCHAR data type

Parameters:

**long-varchar-column**  
The name of a LONG VARCHAR column.

**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.

The SUBSTRING64 function returns a variable length character string of the LONG VARCHAR column parameter. If any of the arguments are NULL, SUBSTRING64 returns NULL.

Syntax:

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

Parameters:

**large-object-column**  
The name of a LONG VARCHAR column.

**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.

For example, given a column named `col1` which 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’
```
The SUBSTRING64 function also supports the LONG BINARY data type.

Nesting of the functions SUBSTRING64, SUBSTRING, BYTE_SUBSTR, and BYTE_SUBSTR64 is not supported.

Aggregate functions

Only the aggregate function COUNT (*) is supported for LONG VARCHAR columns. The COUNT DISTINCT parameter is not supported. An error is returned if a LONG VARCHAR column is used with the MIN, MAX, AVG, or SUM aggregate functions.
Function support of LONG VARCHAR data type
CHAPTER 5

Stored Procedure Support

About this chapter
This chapter describes the stored procedure support for the LONG BINARY (BLOB) and LONG VARCHAR (CLOB) data type columns.

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controlling large object data compression</td>
<td>17</td>
</tr>
<tr>
<td>Displaying information about large object columns</td>
<td>19</td>
</tr>
</tbody>
</table>

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 and allows you 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

<table>
<thead>
<tr>
<th>Function</th>
<th>Sets compression of data in columns of LONG BINARY (BLOB) and LONG VARCHAR (CLOB) data types.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>sp_iqsetcompression (owner, table, column, on_off_flag)</td>
</tr>
<tr>
<td>Permissions</td>
<td>Requires DBA authority.</td>
</tr>
<tr>
<td>Description</td>
<td>The sp_iqsetcompression system stored procedure provides control of compression of LONG BINARY (BLOB) and LONG VARCHAR (CLOB) data type columns. The compression setting only applies to IQ base tables. A side effect of sp_iqsetcompression is that a COMMIT occurs after the compression setting is changed.</td>
</tr>
</tbody>
</table>
Controlling large object data compression

Table 5-1: sp_iqsetcompression parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>Owner of the table for which you are setting compression</td>
</tr>
<tr>
<td>table</td>
<td>Table for which you are setting compression</td>
</tr>
<tr>
<td>column</td>
<td>Column for which you are setting compression</td>
</tr>
<tr>
<td>on_off_flag</td>
<td>Compression setting: ON enables compression, OFF disables compression</td>
</tr>
</tbody>
</table>

Example

For this example, assume the following 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 the sp_iqsetcompression procedure using the following command (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 5-2: sp_iqshowcompression parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>Owner of the table for which you are setting compression</td>
</tr>
<tr>
<td>table</td>
<td>Table for which you are setting compression</td>
</tr>
<tr>
<td>column</td>
<td>Column for which you are setting compression</td>
</tr>
</tbody>
</table>

Example

For this example, assume the following 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 the `sp_iqshowcompression` procedure using the following command (you must have DBA permission):

```sql
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**

The following 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:

<table>
<thead>
<tr>
<th>Username</th>
<th>Indexname</th>
<th>Type</th>
<th>Info</th>
<th>KBytes</th>
<th>Pages</th>
<th>Compressed Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>Total</td>
<td>42953952</td>
<td>623009</td>
<td>622923</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>vdo</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>bt</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>garray</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>bm</td>
<td>136</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>barray</td>
<td>2312</td>
<td>41</td>
<td>40</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>dpstore</td>
<td>170872</td>
<td>2551</td>
<td>2549</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>largelob</td>
<td>42780632</td>
<td>620415</td>
<td>620333</td>
</tr>
</tbody>
</table>

In this example, the compression ratio is \( \frac{42953952}{623009 \times 128} = 53.9\% \).

**Size of a LONG VARCHAR column**

The following 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:

<table>
<thead>
<tr>
<th>Username</th>
<th>Indexname</th>
<th>Type</th>
<th>Info</th>
<th>KBytes</th>
<th>Pages</th>
<th>Compressed Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>Total</td>
<td>42953952</td>
<td>623009</td>
<td>622923</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>vdo</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>bt</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>garray</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>bm</td>
<td>136</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>barray</td>
<td>2312</td>
<td>41</td>
<td>40</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>dpstore</td>
<td>170872</td>
<td>2551</td>
<td>2549</td>
</tr>
<tr>
<td>DBA</td>
<td>test10.DBA.ASIQ_IDX_T128_C3_FP</td>
<td>FP</td>
<td>largelob</td>
<td>42780632</td>
<td>620415</td>
<td>620333</td>
</tr>
</tbody>
</table>

In this example, the compression ratio is \( \frac{42953952}{623009 \times 128} = 53.9\% \).
Displaying information about large object columns
CHAPTER 6

Moving Large Object Data

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

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exporting large object data</td>
<td>21</td>
</tr>
<tr>
<td>Loading large object data</td>
<td>22</td>
</tr>
</tbody>
</table>

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. The BFILE function can be used with or without the data extraction facility.

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 pathname supported by the file system.

**large-object-column**  The name of the LONG BINARY or LONG VARCHAR column.

BFILE returns the following values:

- 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
Loading large object data

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. The SUBSTRING64 or BYTE_SUBSTR64 function must be used to retrieve LONG BINARY cells greater than 2GB using a SELECT (SELECT, OPEN CURSOR). The SUBSTRING64 function must be used to retrieve LONG VARCHAR cells greater than 2GB using a SELECT (SELECT, OPEN CURSOR). Also note that some connection drivers, for example ODBC, JDBC, and Open Client, do not allow more than 2GB to be returned in one SELECT.

**BFILE example**

This example shows the use of the BFILE function to extract data from the LONG BINARY column lobcol, which is created and loaded in the “Load example” on page 23. The following command writes the data in files which can be used as secondary files in a load.

```sql
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

LONG BINARY and LONG VARCHAR data can be loaded using extended syntax of the LOAD TABLE statement. You can 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' } ]
       ... [ 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. Tape devices are not supported for the secondary file. Note that IQ supports loading LONG BINARY and LONG VARCHAR values of length less than or equal to 32767 bytes in the primary load file.

**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 the following 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 `ltab`, 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 the option SECONDARY_FILE_ERROR is ON, the load will rollback, if an error occurs while opening or reading from a secondary BINARY FILE or ASCII FILE.

If the option SECONDARY_FILE_ERROR is OFF, 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 the following value:

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

The allowed values of the SECONDARY_FILE_ERROR option are ON and OFF. The default value is OFF. This option can be set for the PUBLIC group or temporary by any user and 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 multi-byte character data

Partial multi-byte 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 multi-byte 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 multi-byte character according to the value of the SECONDARY_FILE_ERROR database option.
Table 6-1 lists how a trailing multi-byte character is loaded, depending on the values of the TRIM_PARTIAL_MBC and SECONDARY_FILE_ERROR database options.

**Table 6-1: Partial multi-byte character on loading LONG VARCHAR with ASCII FILE option**

<table>
<thead>
<tr>
<th>TRIM_PARTIAL_MBC</th>
<th>SECONDARY_FILE_ERROR</th>
<th>Trailing partial multi-byte character found</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>ON/OFF</td>
<td>Trailing partial multi-byte character truncated</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>Cell — null, if null allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOAD error — rollback, if null not allowed</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>Cell — null, if null allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cell — 0 length, if null not allowed</td>
</tr>
</tbody>
</table>
Loading large object data
Compatibility and Conformance

About this chapter

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

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compatibility</td>
<td>27</td>
</tr>
<tr>
<td>Conformance to standards</td>
<td>27</td>
</tr>
</tbody>
</table>

Compatibility

Adaptive Server Anywhere (ASA) 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 ASA is SQL92 compliant. ASA 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 textual 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 ASE is SQL92 compliant.

Conformance to standards

Sybase IQ LONG BINARY and LONG VARCHAR functionality conforms to the CORE level of the SQL99 standard.
Conformance to standards
CHAPTER 8

Error and Warning Messages

About this chapter

This chapter describes the error and warning messages that may be returned when you are working with LONG BINARY and LONG VARCHAR columns.

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error 1000195</td>
<td>30</td>
</tr>
<tr>
<td>Error 1000198</td>
<td>30</td>
</tr>
<tr>
<td>Error 1001051</td>
<td>31</td>
</tr>
<tr>
<td>Error 1001052</td>
<td>31</td>
</tr>
<tr>
<td>Error 1001053</td>
<td>32</td>
</tr>
<tr>
<td>Error 1001054</td>
<td>32</td>
</tr>
<tr>
<td>Warning 1001055</td>
<td>33</td>
</tr>
<tr>
<td>Warning 1001056</td>
<td>33</td>
</tr>
<tr>
<td>Error 1001057</td>
<td>34</td>
</tr>
<tr>
<td>Error 1001058</td>
<td>34</td>
</tr>
<tr>
<td>Error 1012030</td>
<td>35</td>
</tr>
</tbody>
</table>
Error 1000195

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>-1000195L</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_BINARYFILE</td>
</tr>
<tr>
<td>SQLState</td>
<td>QDB95</td>
</tr>
<tr>
<td>ODBC State</td>
<td>ERROR</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>location of the exception</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>type of load specification</td>
</tr>
<tr>
<td>Parameter 3</td>
<td>data type of column</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>-1000198L</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_CANNOT_CREATE_JOIN_INDEX</td>
</tr>
<tr>
<td>SQLState</td>
<td>QDB98</td>
</tr>
<tr>
<td>ODBC State</td>
<td>ERROR</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>location of the exception</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>data type of column</td>
</tr>
</tbody>
</table>

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).
CHAPTER 8  Error and Warning Messages

Error 1001051
Message text  Query returns %3 data > 2GB. Use %2 %1

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>-1001051L</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_LOB_OVER_2G_W_ARG</td>
</tr>
<tr>
<td>SQLState</td>
<td>QFA47</td>
</tr>
<tr>
<td>ODBC State</td>
<td>ERROR</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>function recommended</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>SA parse source code line</td>
</tr>
<tr>
<td>Parameter 3</td>
<td>long binary or long varchar data type</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>-1001052L</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_ONLY_SUPPORT_LOB_W_ARG</td>
</tr>
<tr>
<td>SQLState</td>
<td>QFA48</td>
</tr>
<tr>
<td>ODBC State</td>
<td>ERROR</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>SA parse source code line</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>LOB argument name</td>
</tr>
<tr>
<td>Parameter 3</td>
<td>recommended function name</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>-1001053L</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_WRONG_NUM_PARAMS_W_ARG</td>
</tr>
<tr>
<td>SQLState</td>
<td>QFA49</td>
</tr>
<tr>
<td>ODBC State</td>
<td>ERROR</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>SA parse source code line</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>function name</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>-1001054L</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_LOB_NOT_ALLOWED_GROUP</td>
</tr>
<tr>
<td>SQLState</td>
<td>QFA50</td>
</tr>
<tr>
<td>ODBC State</td>
<td>ERROR</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>location of the exception</td>
</tr>
</tbody>
</table>

Probable cause  This error is reported when you attempt to use a LONG BINARY column in an ORDER BY, GROUP BY, or aggregation clause.
CHAPTER 8 Error and Warning Messages

Warning 1001055
Message text An error occurred loading %1 column, %2, for %3, rowid %4.

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>1001055L</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_LOB_LOAD_ERROR_WARN</td>
</tr>
<tr>
<td>SQLState</td>
<td>QFA51</td>
</tr>
<tr>
<td>ODBC State</td>
<td>OK</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>long binary or long varchar data type</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>FP index name</td>
</tr>
<tr>
<td>Parameter 3</td>
<td>secondary file name</td>
</tr>
<tr>
<td>Parameter 4</td>
<td>rowid</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>1001056L</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_LOB_EXTRACT_ERROR_WARN</td>
</tr>
<tr>
<td>SQLState</td>
<td>QFA52</td>
</tr>
<tr>
<td>ODBC State</td>
<td>OK</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>long binary or long varchar data type</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>FP index name</td>
</tr>
<tr>
<td>Parameter 3</td>
<td>secondary file name</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>-1001057L</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_LOB_EXTRACT_USE_BFILE</td>
</tr>
<tr>
<td>SQLState</td>
<td>QFA53</td>
</tr>
<tr>
<td>ODBC State</td>
<td>ERROR</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>location of the exception</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>long binary or long varchar data type</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>-1001058L</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_LOB_SECONDARY_FILE_TOOLONG</td>
</tr>
<tr>
<td>SQLState</td>
<td>QFA54</td>
</tr>
<tr>
<td>ODBC State</td>
<td>OK</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>location of the exception</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>secondary file name</td>
</tr>
</tbody>
</table>

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 1012030

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLCode</td>
<td>-1012030</td>
</tr>
<tr>
<td>Constant</td>
<td>EMSG_CAT_PAGESIZETOOSMALL</td>
</tr>
<tr>
<td>SQLState</td>
<td>QUA30</td>
</tr>
<tr>
<td>ODBC State</td>
<td>ERROR</td>
</tr>
<tr>
<td>Parameter 1</td>
<td>location of the exception</td>
</tr>
<tr>
<td>Parameter 2</td>
<td>column number</td>
</tr>
<tr>
<td>Parameter 3</td>
<td>requested page size</td>
</tr>
<tr>
<td>Parameter 4</td>
<td>minimum allowed page size</td>
</tr>
</tbody>
</table>

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.
Error 1012030
APPENDIX A

Upgrading existing LONG BINARY columns

About this appendix

This appendix contains a procedure for upgrading LONG BINARY columns that were created in Sybase IQ 12.5 in versions prior to 12.5 ESD8. If you have LONG BINARY columns created in Sybase IQ 12.5 prior to ESD8, you must follow the first six steps of this procedure before you install Sybase IQ 12.6 or later versions.

Existing LONG BINARY columns created using any Sybase IQ 12.5 release prior to ESD8 are not supported. You must explicitly drop all existing LONG BINARY columns created prior to Sybase IQ 12.5 ESD8 before you install Sybase IQ 12.6 or later versions, then recreate them after you install Sybase IQ 12.6 or later versions. The ALTER DATABASE UPGRADE command of Sybase IQ 12.6 and later versions does not upgrade LONG BINARY columns created prior to Sybase IQ 12.5 ESD8.

Note You do not need to upgrade LONG BINARY columns created in Sybase IQ 12.5 ESD8 or later. Keep in mind, however, that if the server is not licensed for the LOB component, the CREATE TABLE and ALTER TABLE ADD column commands with a LONG BINARY column are not allowed and return the error “Large Objects Management functionality is not licensed on this server.”

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upgrading existing LONG BINARY columns</td>
<td>37</td>
</tr>
</tbody>
</table>

Upgrading existing LONG BINARY columns

Use the following upgrade procedure to extract and drop LONG BINARY columns created prior to the ESD8 level of IQ 12.5, then add LONG BINARY columns and load the extracted data after installing Sybase IQ 12.6 or later versions.
To upgrade existing LONG BINARY columns

1. Before you install Sybase IQ 12.6 or later versions, find all of the existing LONG BINARY columns, using this query:

   ```sql
   SELECT table_name, column_name FROM SYS.SYSTABLE T, SYS.SYSCOLUMN C
   WHERE ((T.table_type='BASE' OR T.table_type='GBL TEMP') AND T.server_type='IQ') AND (T.table_id=C.table_id) AND (C.domain_id=12)
   ```

   As an example, the query result is:

<table>
<thead>
<tr>
<th>table_name</th>
<th>column_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>lb_tab</td>
<td>lbin1</td>
</tr>
<tr>
<td>lb_tab</td>
<td>lbin2</td>
</tr>
</tbody>
</table>

   The table `lb_tab` contains two LONG BINARY columns, `lbin1` and `lbin2`.

2. Add an integer column to store the length in bytes of each LONG BINARY column.

   ```sql
   ALTER TABLE table-name ADD blen-column-name INT
   ```

   where `table-name` is the name of the table that contains the LONG BINARY column and `blen-column-name` is the name of the column to store the byte length of the LONG BINARY column. Use the `table-name` value returned in step 1 and add a byte length column for each LONG BINARY column returned in step 1.

   For example, the statements

   ```sql
   ALTER TABLE lb_tab ADD lbin1_len INT;
   ALTER TABLE lb_tab ADD lbin2_len INT;
   ```

   add new columns to store the byte lengths of the LONG BINARY columns `lbin1` and `lbin2` returned in step 1.
3. Save the byte length of each LONG BINARY column in the newly created length column and commit the update:

```
UPDATE table-name
SET blen-column-name = BYTE_LENGTH( column-name );
COMMIT;
```

where `table-name` is the name of the table that contains the LONG BINARY column, `blen-column-name` is the name of the column to store the byte length of the LONG BINARY column, and `column-name` is the name of the LONG BINARY column.

For example, the statements

```
UPDATE lb_tab
SET lbin1_len = BYTE_LENGTH( lbin1 ),
    lbin2_len = BYTE_LENGTH( lbin2 );
COMMIT;
```

save the byte lengths of the LONG BINARY columns `lbin1` and `lbin2` in the columns `lbin1_len` and `lbin2_len`, respectively.

Repeat this step for every LONG BINARY column that you are upgrading.

4. Extract each LONG BINARY column in each row to a different disk file.

```
SET TEMPORARY OPTION TEMP_EXTRACT_BINARY = 'ON';
SET TEMPORARY OPTION TEMP_NAME1 = 'file-name';
SELECT column-name FROM table-name
WHERE ROWID(table-name) = row-id;
```

where `file-name` is the name of the file to which the LONG BINARY data is extracted, `column-name` is the name of the LONG BINARY column, `table-name` is the name of the table that contains the LONG BINARY column, and `row-id` is the row id of the row in `table-name` that contains the LONG BINARY column.

For example, to extract the LONG BINARY column `lbin1` in row 1 of the example results of step 1, execute the following statements:

```
SET TEMPORARY OPTION TEMP_EXTRACT_BINARY = 'ON';
SET TEMPORARY OPTION TEMP_NAME1 = 'lbExtractl_1.inp';
SELECT lbin1 FROM lb_tab
WHERE ROWID(lb_tab) = 1;
```
Upgrading existing LONG BINARY columns

To extract the LONG BINARY column `lbin1` in row 2 of the example results of step 1, execute the following statements:

```sql
SET TEMPORARY OPTION TEMP_EXTRACT_NAME1 = 'lbExtract1_2.inp';
SELECT lbin1 FROM lb_tab
  WHERE ROWID(lb_tab) = 2;
```

Repeat this step for every LONG BINARY column in every row, for example, the column `lbin2` in row id 1 and row id 2:

```sql
SET TEMPORARY OPTION TEMP_EXTRACT_BINARY = 'ON';
SET TEMPORARY OPTION TEMP_EXTRACT_NAME1 = 'lbExtract2_1.inp';
SELECT lbin2 FROM lb_tab
  WHERE ROWID(lb_tab) = 1;

SET TEMPORARY OPTION TEMP_EXTRACT_NAME1 = 'lbExtract2_2.inp';
SELECT lbin2 FROM lb_tab
  WHERE ROWID(lb_tab) = 2;
```

For more information on the data extraction facility, see “Data extraction options” in Chapter 7, “Moving Data In and Out of Databases” of the Sybase IQ System Administration Guide.

5 Use a text editor to create a primary file as input to the LOAD TABLE command. The primary input file for a LONG BINARY column contains the path of the secondary file, which in this procedure is the extraction output file that contains the LONG BINARY data. Refer to “Loading large object data” on page 22 for more information on using primary and secondary input files to load LONG BINARY data.

In the example of this procedure, use a text editor to create the primary file `lob_lb_Extract.inp`, which contains the following two rows:

```plaintext
lbExtract1_1.inp,lbExtract2_1.inp,
lbExtract1_2.inp,lbExtract2_2.inp,
```

This primary input file specifies the names of the files that contain the data extracted from the columns `lbin1` and `lbin2` in rows 1 and 2 of table `lb_tab`. 
6 Use the ALTER TABLE command to drop the LONG BINARY column.

```sql
ALTER TABLE table-name DROP column-name
```

where `table-name` is the name of the table that contains the LONG BINARY column and `column-name` is the name of the LONG BINARY column.

For example, drop the LONG BINARY columns `lbin1` and `lbin2` from the table `lb_tab`:

```sql
ALTER TABLE lb_tab DROP lbin1;
ALTER TABLE lb_tab DROP lbin2;
```

Repeat this step for every LONG BINARY column after the LONG BINARY data has been extracted from the column.

7 After you install Sybase IQ 12.6 or later versions with the LOB component and run the ALTER DATABASE UPGRADE command as part of the installation procedure, use the ALTER TABLE command to add the LONG BINARY column.

```sql
ALTER TABLE table-name ADD column-name LONG BINARY
```

where `table-name` is the name of the table that contains the LONG BINARY column and `column-name` is the name of the LONG BINARY column.

**Note** Running the ALTER DATABASE UPGRADE command as part of the Sybase IQ 12.6 or later installation is required for using the domains BLOB and CLOB.

For example, add the LONG BINARY columns `lbin1` and `lbin2` to the table `lb_tab`:

```sql
ALTER TABLE lb_tab ADD lbin1 LONG BINARY;
ALTER TABLE lb_tab ADD lbin2 LONG BINARY;
```

Repeat this step for every LONG BINARY column that you are upgrading.

8 Use the LOAD TABLE command to load the extracted data into the LONG BINARY column, then commit the transaction. Refer to “Loading large object data” on page 22 for the necessary LOAD TABLE syntax.
For example, load the extracted data back into the LONG BINARY columns lbin1 and lbin2:

```sql
LOAD TABLE lb_tab
  ( lbin1 BINARY FILE (','),
    lbin2 BINARY FILE (',') )
FROM 'lob_lbExtract.inp'
QUOTES OFF ESCAPES OFF
ROW DELIMITED BY '\n'
START ROW ID 1;
COMMIT;
```

Repeat this step for every LONG BINARY column that you are upgrading.

9 Update the LONG BINARY column with the correct length, then commit the update:

```sql
UPDATE table-name
SET column-name = BYTE_SUBSTR( column-name, 1, blen-column-name );
COMMIT;
```

where `table-name` is the name of the table that contains the LONG BINARY column you loaded in step 8, `column-name` is the name of the LONG BINARY column, and `blen-column-name` is the name of the column that contains the byte length of the LONG BINARY column (saved in step 3).

For example, update the lengths of the LONG BINARY columns lbin1 and lbin2:

```sql
UPDATE lb_tab SET
  lbin1 = BYTE_SUBSTR( lbin1, 1, lbin1_len ),
  lbin2 = BYTE_SUBSTR( lbin2, 1, lbin2_len );
COMMIT;
```

Repeat this step for every LONG BINARY column that you are upgrading.

For more information on the `BYTE_SUBSTR` function, see “`BYTE_SUBSTR64 and BYTE_SUBSTR` functions” on page 11.
10 Check the data you have loaded into the LONG BINARY column.

For example, the following query checks the data in the LONG BINARY column `lbin1` in table `lb_tab`:

```sql
SELECT BYTE_LENGTH(lbin1), lbin1_len,
       BYTE_SUBSTR(lbin1,1,20),
       BYTE_SUBSTR(lbin1,(lbin1_len - 20),21)
FROM lb_tab
```

The query result is:

```
64800 64800 AaPp1AaPp2AaPp3AaPp4 1fFuU2fFuU3fFuU4fFuU5
64800 64800 AaPp1AaPp2AaPp3AaPp4 1fFuU2fFuU3fFuU4fFuU5
```

11 Use the ALTER TABLE command to drop the byte length columns:

```
ALTER TABLE table-name DROP blen-column-name
```

where `table-name` is the name of the table that contains the reloaded LONG BINARY columns and `blen-column-name` is the name of the column that contains the length of the LONG BINARY column.

For example, drop the byte length columns `lbin1_len` and `lbin2_len`:

```
ALTER TABLE lb_tab DROP lbin1_len;
ALTER TABLE lb_tab DROP lbin2_len;
```

The extracted and loaded LONG BINARY columns are now upgraded and ready to use.
Upgrading existing LONG BINARY columns
Index

A
Adaptive Server Enterprise
  inserting IMAGE data  4
  inserting TEXT data  8

B
BFILE function  21
  example  22
  extraction example  22
  extraction facility  21
  syntax  21
binary large object
  aggregate function support  12
BLOB  3
  BYTE_LENGTH64 function  11
  BYTE_SUBSTR function  11
  BYTE_SUBSTR64 function  11
  columns  4
  data type  3
  data type conversions  4
  description  3
  host variables  5
  in queries  5
  indexes  4
  inserting IMAGE data  4
  loading data  22
  LONG BINARY  3
  modifying  4
  monitoring performance  5
  size  4
  sp_iqindexsize  19
  stored procedure support  17
BYTE_LENGTH64 function  11
BYTE_SUBSTR function  11
BYTE_SUBSTR64 function  11

C
CHAR_LENGTH function
  description  13
  syntax  13
CHAR_LENGTH64 function
  description  13
  syntax  13
character large object
Index

aggregate function support 15
BIT_LENGTH function 12
CHAR_LENGTH function 13
CHAR_LENGTH64 function 13
CLOB 7
columns 7
data type 7
data type conversions 8
description 7
host variables 9
in queries 9
indexes 8
inserting TEXT data 8
LONG VARCHAR 7
modifying 8
monitoring performance 9
OCTET_LENGTH function 13
size 7
sp_iqindexsize 19
stored procedure support 17
SUBSTRING function 13
SUBSTRING64 function 14
compatibility
with Adaptive Server Anywhere 27
with Adaptive Server Enterprise 27
with ASA 27
with ASE 27
compression of LOB data 17
changing settings 17
displaying settings 18
conventions
documentation ix, x
syntax ix
typographic x

D
data compression of LOB 17
changing settings 17
displaying settings 18
data type
BLOB 3
CLOB 7
LONG BINARY 3
LONG VARCHAR 7
data type conversion
LONG BINARY to BINARY 4
LONG BINARY to VARBINARY 4
LONG VARCHAR to CHAR 8
LONG VARCHAR to VARCHAR 8
databases
sample xi
disabling compression 17
documentation
accessibility features xi
Adaptive Server Anywhere vii
conventions ix, x
on CD vii
online vii
Sybase IQ vi

E
enabling compression 17
<table>
<thead>
<tr>
<th>Error Messages</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exporting</td>
<td></td>
</tr>
<tr>
<td>BFILE example</td>
<td>22</td>
</tr>
<tr>
<td>BFILE function</td>
<td>21</td>
</tr>
<tr>
<td>BLOB</td>
<td>21</td>
</tr>
<tr>
<td>CLOB</td>
<td>21</td>
</tr>
<tr>
<td>Large Object Data</td>
<td>21</td>
</tr>
<tr>
<td>LOB</td>
<td>21</td>
</tr>
<tr>
<td>LONG BINARY</td>
<td>21</td>
</tr>
<tr>
<td>Exporting LONG VARCHAR</td>
<td>21</td>
</tr>
<tr>
<td>Extraction Facility</td>
<td></td>
</tr>
<tr>
<td>BFILE function</td>
<td>21</td>
</tr>
</tbody>
</table>

### F

Federal Rehabilitation Act

Section 508 xi

Functions

- BFILE 21
- BFILE example 22
- BIT_LENGTH 12
- BYTE_LENGTH64 11
- BYTE_SUBSTR 11
- BYTE_SUBSTR64 11
- CHAR_LENGTH 13
- CHAR_LENGTH64 13
  - for BLOB 11
  - for CLOB 12
  - for LOB 11
- LONG BINARY aggregate support 12
- LONG VARCHAR aggregate support 15
- OCTET_LENGTH 13
- SUBSTRING 13
- SUBSTRING64 14

### H

Host Variables

- Binary large object 5
- BLOB 5
- Character large object 9
- CLOB 9
- LONG BINARY 5
- LONG VARCHAR 9

## Large Objects Management

### I

**IMAGE Data**

- Inserting from ASE 4
- Inserting into LONG BINARY 4

**Indexes**

- Binary large object 4
- BLOB 4
- Character large object 8
- CLOB 8
- LONG BINARY 4
- LONG VARCHAR 8

**Installing**

- Upgrade prior to 3, 37

### L

**Large Object Data**

- Exporting 21
- Loading 22

**Licensing**

- 1

**LOAD TABLE**

- Example 23
  - Extended syntax 23
  - Primary load file 23
  - Secondary load file 23

**Loading**

- BLOB 22
- CLOB 22
  - Controlling errors 24
  - Large object data 22
  - LOAD TABLE example 23
- LOB 22
- LONG BINARY 22
  - SECONDARY_FILE_ERROR option 24
  - Stripping trailing blanks 24
  - TRIM_PARTIAL_MBC option 24
  - Truncating character data 24

**Loading LONG VARCHAR**

- 22

**LOB**

- Exporting data 21
  - Function support 21
  - Introduction 1
  - Loading data 22
  - Typical sources 1
  - LOB compression
Index

changing settings 17
disabling 17
displaying settings 18
enabling 17
LOB option
  licensing 1
LONG BINARY
  aggregate function support 12
  binary large object 3
  BLOB 3
  BYTE_LENGTH64 function 11
  BYTE_SUBSTR function 11
  BYTE_SUBSTR64 function 11
  columns 4
  data type conversions 4
  exporting data 21
  host variables 5
  in queries 5
  indexes 4
  inserting IMAGE data 4
  loading data 22
  modifying 4
  monitoring performance 5
  size 4
  sp_iqindexsize 19
  stored procedure support 17
  upgrading existing columns 3, 37
  upgrading from 12.5 3, 37
LONG VARCHAR
  aggregate function support 15
  BIT_LENGTH function 12
  CHAR_LENGTH function 13
  CHAR_LENGTH64 function 13
  character large object 7
  CLOB 7
  columns 7
  data type conversions 8
  exporting data 21
  host variables 9
  in queries 9
  indexes 8
  inserting TEXT data 8
  loading data 22
  modifying 8
  monitoring performance 9
  OCTET_LENGTH function 13
  size 7
  sp_iqindexsize 19
  stored procedure support 17
  SUBSTRING function 13
  SUBSTRING64 function 14

M
  messages
    error 29
    warning 29
  multi-byte characters
    TRIM_PARTIAL_MBC option 24
    trimming partial 24
    truncating on load 24

O
  OCTET_LENGTH function
    description 13
    syntax 13

P
  performance monitor
    binary large object 5
    BLOB 5
    character large object 9
    CLOB 9
    LONG BINARY 5
    LONG VARCHAR 9

Q
  queries
    binary large object 5
    BLOB 5
    character large object 9
    CLOB 9
    LONG BINARY 5
    LONG VARCHAR 9
S
sample database xi
SECONDARY_FILE_ERROR option 24
section 508
    compliance xi
sp_iqindexsize
    binary large object 19
    BLOB 19
    character large object 19
    CLOB 19
    LONG BINARY 19
    LONG VARCHAR 19
sp_iqindexsize stored procedure 19
sp_iqsetcompression stored procedure 17
sp_iqshowcompression stored procedure 18
standards 27
    section 508 compliance xi
standards and compatibility
    section 508 compliance xi
stored procedures
    binary large object 17
    BLOB 17
    character large object 17
    CLOB 17
    LONG BINARY 17
    LONG VARCHAR 17
    sp_iqindexsize 19
    sp_iqsetcompression 17
    sp_iqshowcompression 18
STRING_RTRUNCATION option 5, 8, 9
SUBSTRING function
    description 13
    syntax 13
SUBSTRING64 function
    description 14
    syntax 14

T
TEXT data
    inserting from ASE 8
    inserting into LONG VARCHAR 8
TRIM_PARTIAL_MBC option 24

U
upgrading
    existing LONG BINARY columns 3, 37
    LONG BINARY 3, 37

W
warning messages 29

Index

Large Objects Management 49