

# **Technical Bulletin**

**Part No. 74-0115**

## **DataStage Sybase OC**

This technical bulletin describes Release 2.1 of the DataStage Sybase OC stage. This stage includes a GUI to read and write data to and from a Sybase database using the Sybase Open Client Client-Library interface.

Copyright © 2003 Ascential Software Corporation  
50 Washington Street, Westboro, MA 01581  
All rights reserved.

© 1998–2003 Ascential Software Corporation. All rights reserved. Ascential, Ascential Software, DataStage, MetaStage, MetaBroker, and Axielle are trademarks of Ascential Software Corporation or its affiliates and may be registered in the United States or other jurisdictions. Adobe Acrobat is a trademark of Adobe Systems, Inc. Microsoft, Windows, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Adaptive Server, Open Client, and Sybase are either registered trademarks or trademarks of Sybase, Inc. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd. Other marks mentioned are the property of the owners of those marks.

This product may contain or utilize third party components subject to the user documentation previously provided by Ascential Software Corporation or contained herein.

## **Printing History**

First Edition (74-0115) for Release 1.0, February 1998  
Updated (74-0115) for Release 1.0.1, March 1998  
Second Edition (74-0115) for Release 1.1, April 1998  
Updated (74-0115) for Release 1.0.2, May 1998  
Third Edition (74-0115) for Release 1.2, June 1998  
Fourth Edition (74-0115) for Release 2.0, October 1998  
Updated (74-0115) for Release 2.0.1, February 1999  
Fifth Edition (74-0115) for Release 2.1, March 1999  
Updated for Release 2.1, February 2002  
Updated for Release 2.1, August 2002  
Updated for Release 2.1, August 2003

## **How to Order Technical Documents**

To order copies of documents, contact your local Ascential subsidiary or distributor, or call our main office at (508) 366-3888.

Documentation Team: Marie E. Hedin

## Introduction

This technical bulletin describes the following for Release 2.1 of DataStage Sybase OC for DataStage Release 7.0:

- Functionality
- Installation
- Defining the Sybase OC connection
- Defining character set mapping
- Defining an input link
- Defining an output link
- Sybase SQL Server data type support
- Stored procedure support
- Stage, input, and output properties

Sybase Open Client allows C and C++ applications to connect to and process SQL statements in the Sybase SQL Server environment. Sybase OC enables DataStage to read and write data to and from a Sybase database using the Sybase Open Client Client-Library interface.

Each Sybase OC stage is a passive stage that can have any number of the following links:

- **Input links.** Specify the data you are writing, which is a stream of rows to be loaded into a Sybase database.
- **Output links.** Specify the data you are extracting, which is a stream of rows to be read from a Sybase database. You can specify the data on an input or output link using an SQL statement constructed by DataStage or a user-defined query.
- **Reference output link.** Each represents rows that are key read from a Sybase database (using the key columns in a WHERE clause of the SELECT statement that is constructed by DataStage or specified by the user).

For more information on using a plug-in stage in a DataStage job, see *DataStage Server Job Developer's Guide*. For SQL syntax information, see *Sybase Transact-SQL User's Guide*.

## Functionality

Sybase OC has the following functionality:

- Supports stream input, stream output, and reference output links.
- Provides faster access to Sybase tables than the ODBC stage by directly accessing the Sybase Open Client interface.
- Automatically generates SQL statements to read or write Sybase data (you can override these with user-defined SQL statements).
- Imports table and column definitions.
- Browses Sybase source or target data using the GUI.
- Supports MetaStage. For information, see *MetaStage User's Guide*.
- Supports reject row handling. For information, see DataStage documentation.
- Supports NLS (National Language Support). For information, see *DataStage NLS Guide*.

The following functionality is not supported:

- Bulk loading of Sybase tables from stream input links. Use the BCPLoad stage to do this.
- *binary*, *varbinary*, *text*, *image*, and *timestamp* Sybase data types.

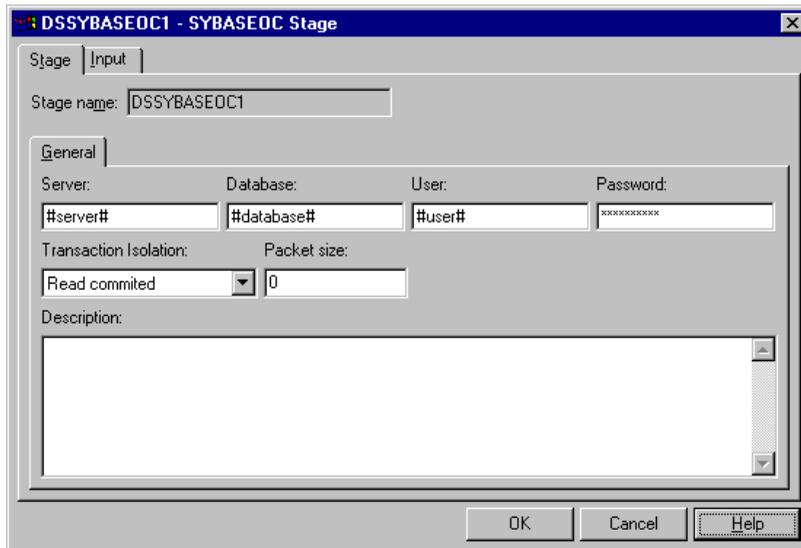
## Installing the Plug-In

For instructions and information supporting the installation, see *DataStage Plug-In Installation and Configuration Guide*.

**Note:** If you experience any timeout issues, increase the default values for the Sybase configuration parameters CS\_RETRY\_COUNT and CS\_TIMEOUT\_VALUE to 10 or higher.

## Defining the Sybase OC Stage

To edit a Sybase OC stage, the SYBASEOC Stage dialog box appears:



This dialog box has the following pages (depending on whether there are inputs to and outputs from the stage):

- **Stage.** This page displays the name of the stage you are editing. The **General** tab defines the Sybase server, database, login information, transaction isolation level, and packet size information for concurrency control and performance tuning in jobs. You can connect to a Sybase database. You can describe the purpose of the stage in the **Description** field on the **General** tab. For details, see [“Connecting to a Sybase Database”](#) on page 4.

The **NLS** tab defines a character set map to be used with the stage. (This tab appears only if you have installed NLS.) For details, see [“Defining Character Set Mapping”](#) on page 5.

- **Input.** This page is displayed only if you have an input link to this stage. It specifies the SQL table to use and the associated column definitions for each data input link. This page also specifies how data is written and contains the SQL statement or call syntax used to write data to a Sybase table.
- **Output.** This page is displayed only if you have an output or reference link to this stage. It specifies the SQL tables to use and the associated column

definitions for each data output link. It also contains the SQL SELECT statement or call syntax used to read data from one or more Sybase tables or views.

The main phases in defining the Sybase OC stage from the **SYBASEOC Stage** dialog box are:

1. Connect to an Sybase database (see the next topic).
2. Optionally define a character set map (see [page 5](#)).
3. Define the data on the input links (see [page 6](#)).
4. Define the data on the output links (see [page 12](#)).

## Connecting to a Sybase Database

The Sybase OC connection parameters are set on the **General** tab of the **Stage** page. To connect to a Sybase database:

1. Enter the name of the system where the Sybase server is installed in the **Server** field. This name should correspond to an entry in the Sybase sql.ini (Windows clients) or *interfaces* (UNIX clients) file that contains the server information. This field is required. There is no default.
2. Enter the name of the Sybase database name to access in the **Database** field. Unless the database has a guest account, **User** must be a valid user in the database, have an alias in the database, or be a system administrator or system security officer. If you do not specify **Database**, the default database for the user (as configured in Sybase) is accessed.
3. Enter the name to use to connect to the Sybase server in the **User** field. This user must have sufficient privileges to access the specified database and source and target tables. This field is required. There is no default.
4. Enter the password that is associated with the specified user name to use in the **Password** field. There is no default.
5. Choose an appropriate transaction isolation level to use from the **Transaction Isolation** list. This level provides the necessary concurrency control between transactions in the job and other transactions.

6. Use one of the following transaction isolation levels:

**Level 0 (read uncommitted).** Takes exclusive locks on modified data. These locks are held until a commit or rollback is executed. However, other transactions can still read (but not modify) the uncommitted changes. No other locks are taken.

**Level 1 (read committed).** Takes exclusive locks on modified data and sharable locks on all other data. Exclusive locks are held until a commit or rollback is executed. Uncommitted changes are not readable by other transactions. Shared locks are released immediately after the data has been processed, allowing other transactions to modify it.

**Level 3 (serializable).** Takes exclusive locks on modified data and sharable locks on all other data. All locks are held until a commit or rollback is executed, preventing other transactions from modifying any data that has been referenced during the transaction.

7. Enter the packet size for Sybase client/server communication in the **Packet size** field. The correct setting of this property can improve performance.

The value must be a multiple of 512 and cannot exceed the “max network packet size” Sybase server parameter. If the specified value is not a multiple of 512, it is rounded down to the nearest multiple. If no value is specified, the “default network packet size” server parameter value is used.

To increase the “max network packet size” parameter or check its current value, use the Sybase *sp\_configure* system procedure. You must also increase the “additional network memory” server parameter to accommodate any increase in “max network packet size.” After changing these parameters, you must restart the Sybase server so the new values take effect.

For more information on setting server parameters, see “Setting Configuration Parameters” in *Sybase SQL Server System Administration Guide*. For information on the performance advantages of increasing the “max network packet size” server parameter, see “Networks and Performance” in *Sybase Performance and Tuning Guide*.

8. Optionally describe the Sybase OC stage in the **Description** field.

## Defining Character Set Mapping

You can define a character set map for a stage from the NLS tab that appears on the **Stage** page. The NLS tab only appears if you have installed NLS.

The default character set map is defined for the project or the job. You can change the map by selecting a map name from the **Map name to use with stage** list.

Click **Use Job Parameter...** to specify parameter values for the job. Use the format `#Param#`, where *Param* is the name of the job parameter. The string `#Param#` is replaced by the job parameter when the job is run.

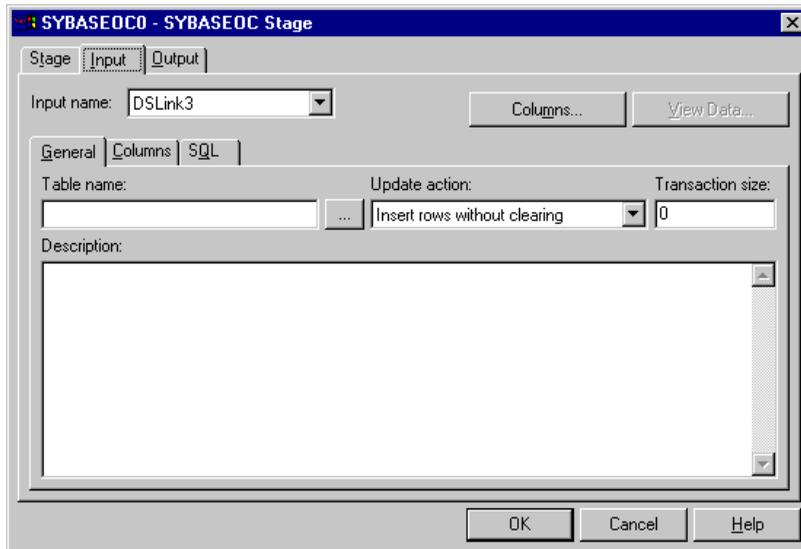
Select **Show all maps** to list all the maps that are shipped with DataStage.

Select **Loaded maps only** to list only the maps that are currently loaded.

For more information about NLS or job parameters, see DataStage documentation.

## Defining Sybase Input Data

When you write data to a table in a Sybase database, the Sybase OC stage has an input link. The properties of this link and the column definitions of the data are defined on the **Input** page in the **SYBASEOC Stage** dialog box.



### About the Input Page

The **Input** page has one field, three tabs, and two buttons:

- **Input name.** The name of the input link. Choose the link you want to edit from the **Input name** list. This list displays all the input links to the Sybase OC stage.

- **General.** This tab is displayed by default. It contains the following parameters:
  - **Table name.** This field is editable when the update action is *not* **User defined SQL** (otherwise, it is read-only). Depending on the update action selected, the user name specified on the **Stage** page must have insert, update, or delete privileges on the table named in **Table name**.

It is the name of the target table the data is written to. You must specify **Table name** if you do not specify **User defined SQL**. There is no default.

You can also click ... to browse the Repository to select the table.

- **Update action.** Specifies which SQL statements are used to update the target table. Some update actions require key columns to update or delete rows. There is no default. Choose the option you want from the list:

**Clear table then insert rows.** Clears the table using the delete command and inserts new rows. Delete is equivalent to, but slower than, truncate table. (Delete removes rows one at a time and logs each deleted row as a transaction. Truncate table deallocates whole data pages and makes fewer log entries.)

**Truncate table then insert rows.** Clears the table using the truncate table command and inserts new rows. Truncate table is equivalent to, but faster than, delete. (Delete removes rows one at a time and logs each deleted row as a transaction. Truncate table deallocates whole data pages and makes fewer log entries.)

**Insert rows without clearing.** Inserts new rows in the table.

**Delete existing rows only.** Deletes the existing rows in the target file that have identical keys in the source files.

**Replace existing rows completely.** Deletes the existing rows, then adds the new rows to the table.

**Update existing rows only.** Updates the existing data rows. Any rows in the data that do not exist in the table are ignored.

**Update existing rows or insert new rows.** Updates the existing data rows before adding new rows. Performance depends on the contents of the target table and the rows being processed in the job. If most rows exist in the target table, it is faster to update first.

**Insert new rows or update existing rows.** Inserts the new rows before updating existing rows. Performance depends on the contents of the

target table and the rows being processed in the job. If most rows do not exist in the target table, it is faster to insert first.

**User defined SQL.** Writes the data using a user-defined SQL statement. When you select this option, it overrides the default SQL statement generated by the stage.

- **Transaction size.** The number of rows that DataStage processes before committing a transaction to the database. The default value of 100 is recommended for optimal performance. If set to 0, DataStage commits one transaction after all rows are processed. Setting this property to 0 or a large nonzero value requires Sybase to maintain very large open transactions, which adds overhead and can decrease performance. Setting this property to a small nonzero value results in frequent transaction commits, which also adds overhead and can decrease performance.
- **Description.** Contains an optional description of the input link.
- **Columns.** This tab contains the column definitions for the data written to the table or file. The column definitions are used in the order they appear in the Columns grid. The **Columns** tab behaves the same way as the **Columns** tab in the ODBC stage. For a description of how to enter and edit column definitions, see *DataStage Designer Guide*.
- **SQL.** This tab contains one field and four tabs:
  - **Input name.** The name of the input link. Choose the link you want to edit from the **Input name** list. This list displays all the input links to the Sybase OC stage.
  - **Generated.** This tab is displayed by default. It contains the SQL statements constructed by DataStage that are used to write data to Sybase. You cannot edit these statements, but you can use **Copy** to copy them to the Clipboard for use elsewhere. See [“Using Generated SQL Statements”](#) on page 9.
  - **User-defined.** This tab contains the SQL statements executed to write data to Sybase. See [“Using User-Defined SQL Statements”](#) on page 10.
  - **Before.** This tab contains the SQL statements executed before the stage processes any job data rows. See [“Using BeforeSQL Statements”](#) on page 11.
  - **After.** This tab contains the SQL statements executed after the stage processes job data rows. See [“Using AfterSQL Statements”](#) on page 11.

Click **Columns...** to display a brief list of columns designated on the input link. As you enter detailed meta data in the **Columns** tab, you can leave this list displayed.

Click **View Data...** to start the Data Browser. This lets you look at the data associated with the input link. For a description of the Data Browser, see DataStage documentation.

## Writing Data to Sybase

The following sections describe the differences when you use generated or user-defined SQL INSERT, DELETE, or UPDATE statements to write data from DataStage to a Sybase database.

### Using Generated SQL Statements

By default, DataStage writes data to a Sybase table using an SQL INSERT, DELETE, or UPDATE statement that it constructs. The generated SQL statement is automatically constructed using the DataStage table and column definitions that you specify in the input properties for this stage. The **Generated** tab on the **SQL** tab displays the SQL statement used to write the data.

To use a generated statement:

1. Enter a table name in the **Table name** field on the **General** tab of the **Input** page.
2. Specify how you want the data to be written by choosing an option from the **Update action** list:
  - **Clear table then insert rows**
  - **Truncate table then insert rows**
  - **Insert rows without clearing**
  - **Delete existing rows only**
  - **Replace existing rows completely**
  - **Update existing rows only**
  - **Update existing rows or insert new rows**
  - **Insert new rows or update existing rows**
  - **User defined SQL**

See [“Defining Sybase Input Data”](#) on page 6 for a description of each update action.

3. Enter an optional description of the input link in the **Description** field.

4. Click the **Columns** tab on the **Input** page.
5. Edit the Columns grid to specify column definitions for the columns you want to write.

The SQL statement is automatically constructed using your chosen update action and the columns you have specified. You can now optionally view this SQL statement.

6. Click the **SQL** tab on the **Input** page, then the **Generated** tab to view this SQL statement. You cannot edit the statement here, but you can access this tab at any time to select and copy parts of the generated statement to paste into the user-defined SQL statement.
7. Click **OK** to close the **SYBASEOC Stage** dialog box. Changes are saved when you save your job design.

## Using User-Defined SQL Statements

Instead of writing data using an SQL statement constructed by DataStage, you can enter your own SQL INSERT, DELETE, or UPDATE statement for each Sybase OC input link. Ensure that the SQL statement contains the table name, the type of update action you want to perform, and the columns you want to write.

To enter an SQL statement:

1. Choose **User defined SQL** from the **Update action** list on the **General** tab of the **Input** page.
2. Click the **User-defined** tab on the **SQL** tab. The **User-defined SQL** tab appears.

Enter the SQL statement you want to use or edit to write data to the target Sybase tables. This statement must contain the table name, the type of update action you want to perform, and the columns you want to write.

If the property value begins with {FILE}, the remaining text is interpreted as a pathname, and the contents of the file supplies the property value.

When writing data, the INSERT statements must contain a VALUES clause with a parameter marker ( ? ) for each stage input column. UPDATE statements must contain a SET clause with parameter markers for each stage input column. UPDATE and DELETE statements must contain a WHERE clause with parameter markers for the primary key columns. The parameter markers must be in the same order as the associated columns listed in the stage properties. For example:

```
insert emp (emp_no, emp_name) values (?, ?)
```

If you specify multiple SQL statements, they are executed as one or more Transact-SQL command batches using a semicolon ( ; ) as the end-of-batch signal. You cannot combine multiple INSERT, UPDATE, and DELETE statements in one batch. You must execute each in a separate command batch.

You cannot call stored procedures as there is no facility for passing the row values as parameters. (You can call stored procedures for output.)

Unless you specify a user-defined SQL statement, the stage automatically generates an SQL statement.

3. Click **OK** to close the **SYBASEOC Stage** dialog box. Changes are saved when you save your job design.

### Using BeforeSQL Statements

You can execute SQL statements before the stage processes any job data rows. To specify SQL statements before processing any data:

1. Enter the SQL statements you want to be executed before data is processed in the text entry area on the **Before** tab of the **SQL** tab.

If the property value begins with {FILE}, the remaining text is interpreted as a pathname and the content of the file supplies the property value.

Execution occurs immediately after a successful Sybase client/server connection. If you specify multiple SQL statements, they are executed as one or more Transact-SQL command batches using a semicolon ( ; ) as the end-of-batch signal.

2. Select the **Treat errors as non-fatal** check box to log BeforeSQL execution errors as warnings. Processing continues with the next command batch, if any. Each successful execution is committed as a separate transaction.

If this check box is cleared, BeforeSQL execution errors are considered fatal to the job and result in a transaction rollback. The transaction is committed only if all BeforeSQL statements successfully execute.

### Using AfterSQL Statements

You can execute SQL statements after the stage processes all job data rows. To specify SQL statements after processing data:

1. Enter the SQL statements you want to be executed after the data is processed in the text entry area on the **After** tab.

If the property value begins with {FILE}, the remaining text is interpreted as a pathname and the content of the file supplies the property value.

Execution occurs immediately before the successful Sybase client/server connection is terminated. If you specify multiple SQL statements, they are executed as one or more Transact-SQL command batches using a semicolon (;) as the end-of-batch signal.

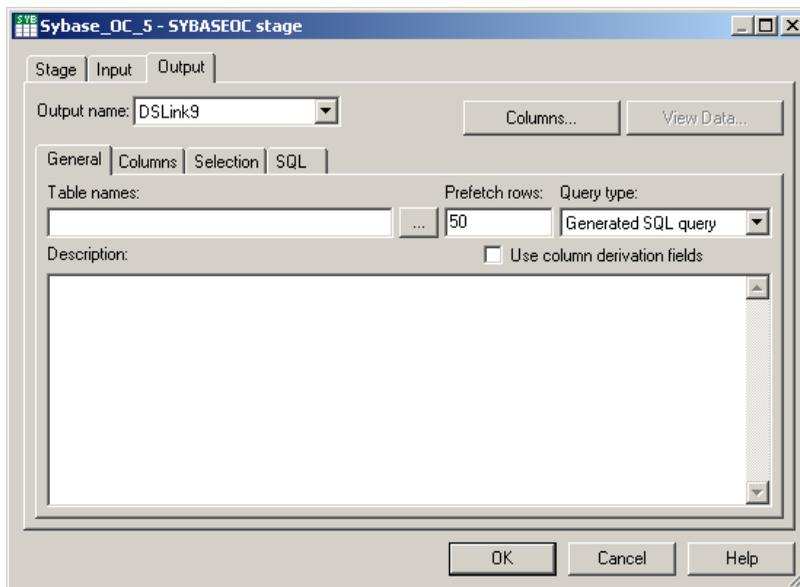
2. Select the **Treat errors as non-fatal** check box to log AfterSQL execution errors as warnings. Processing continues with the next command batch, if any. Each successful execution is committed as a separate transaction.

If this check box is cleared, AfterSQL execution errors are considered fatal to the job and result in a transaction rollback. The transaction is committed only if all AfterSQL statements successfully execute.

## Defining Sybase Output Data

When you read data from a Sybase data source, the Sybase OC stage has an output link.

The properties of the output link and the column definitions of the data are defined on the **Output** page in the **SYBASEOC Stage** dialog box.



## About the Output Page

The **Output** page has one field, up to four tabs, and two buttons. The tabs displayed depend on how you choose to specify the SQL statement to output the data.

- **Output name.** The name of the output link. Choose the link you want to edit from the **Output name** list. This list displays all the output links from the Sybase OC stage.
- **General.** This tab is displayed by default. It contains the following parameters:
  - **Table names.** This field appears only when you select **Generated SQL query**. It contains the names of the Sybase source tables or files being accessed. These tables must exist or be created and populated by the BeforeSQL statements. Separate multiple table names by a comma ( , ). You must have select privileges on each table. There is no default.

If you specify the query type as **User-defined SQL query**, **Table names** is ignored. You must specify **Table names** if you do not define the query type as **User-defined SQL query**.

You can click ... to browse the Repository to select tables.

Additionally, you can use a job parameter to specify the table name. For details on how to use define and use job parameters, see *DataStage Designer's Guide*.

- **Prefetch rows.** The number of rows that Sybase returns when DataStage fetches data from the source tables. Specifying a value greater than 1 improves performance (memory usage increases to accommodate buffering multiple rows). The stage uses this property to bind fetch buffer arrays.

For more information, see “Array Binding” in *Sybase Open Client Client-Library/C Reference Manual*.

- **Query type.** Displays the following options:

**Generated SQL query.** This is the default setting, which specifies that the data is extracted using an SQL statement constructed by DataStage. When this option is selected, the **Generated** tab appears on the **SQL** tab. You cannot edit this statement.

**User-defined SQL query.** Specifies that the data is extracted using a user-defined SQL query. When this option is selected, the **User-defined** tab appears on the **SQL** tab allowing you to edit SQL statements.

- **Use column derivation fields.** Specifies that the column derivation field is to be used when generating the SQL SELECT statement. If **Use column derivation fields** is selected, the column derivation field is used. If **Use column derivation fields** is not selected (the default), the column derivation field is not used.
- **Description.** Contains an optional description of the output link.
- **Columns.** This tab contains the column definitions for the data being output on the chosen link. For a description of how to enter and edit column definitions, see *DataStage Designer's Guide*. This tab also specifies which columns are aggregated.

The column definitions for output and reference links contain a key field. Key fields are used to join primary and reference inputs to a Transformer stage. The Sybase OC plug-in key reads the data by using a WHERE clause in the SQL SELECT statement. For details on how key fields are specified and used, see DataStage documentation.

- **Selection.** This tab is used primarily with generated SQL statements. It contains optional SQL SELECT clauses for the conditional extraction of data.
- **SQL.** This tab displays the SQL statements or stored procedure call syntax used to read data from Sybase. It contains one field and four tabs:
  - **Generated.** This tab is displayed by default. It contains the SQL statements constructed by DataStage. You cannot edit these statements, but you can use **Copy** to copy them to the Clipboard for use elsewhere. See [“Using Generated Queries”](#) on page 15.
  - **User-defined.** This tab contains the SQL statements executed to write data to Sybase. This tab is enabled when you select **User-defined SQL query** from the **General** tab of the **Output** page. See [“Using User-Defined Queries”](#) on page 16.
  - **Before.** This tab contains the SQL statements executed before the stage processes any job data rows. See [“Using BeforeSQL Statements”](#) on page 11.
  - **After.** This tab contains the SQL statements executed after the stage processes all job data rows. See [“Using AfterSQL Statements”](#) on page 11.

Click **Columns...** to display a brief list of columns designated on the output link. As you enter detailed meta data in the **Columns** tab, you can leave this list displayed.

Click **View Data...** to start the Data Browser. This lets you look at the data associated with the output link. For a description of the Data Browser, see DataStage documentation.

## Reading Data from Sybase

The following sections describe the differences when you use generated queries or user-defined queries to read data from a Sybase database into DataStage.

The column definitions for reference links must contain a key field. You use key fields to join primary and reference inputs to a Transformer stage. The Sybase OC plug-in key reads the data by using a WHERE clause in the SQL SELECT statement.

### Using Generated Queries

By default, DataStage extracts data from a Sybase OC data source using an SQL SELECT statement that it constructs. The SQL statement is automatically constructed using the table and column definitions that you entered on the **Output** page.

When you select **Generated SQL query**, data is extracted from a Sybase database using an SQL SELECT statement constructed by DataStage. SQL SELECT statements have the following syntax:

```
SELECT clause FROM clause
    [WHERE clause]
    [GROUP BY clause]
    [HAVING clause]
    [ORDER BY clause];
```

When you specify the tables to use and the columns to be output from the Sybase OC stage, the SQL SELECT statement is automatically constructed and can be viewed by clicking the **SQL** tab on the **Output** page.

For example, if you extract the columns **Name**, **Address**, and **Phone** from a table called Table1, the SQL statement displayed on the **SQL** tab is:

```
SELECT Name, Address, Phone FROM Table1;
```

The SELECT and FROM clauses are the minimum required and are automatically generated by DataStage. However, you can use any of these SQL SELECT clauses:

SELECT clause	Specifies the columns to select from the database.
FROM clause	Specifies the tables containing the selected columns.

WHERE clause	Specifies the criteria that rows must meet to be selected.
GROUP BY clause	Groups rows to summarize results.
HAVING clause	Specifies the criteria that grouped rows must meet to be selected.
ORDER BY clause	Sorts selected rows.

If you want to use the additional SQL SELECT clauses, you must enter them on the **Selection** tab on the **Output** page.

The **Selection** tab is divided into two areas (panes). You can resize an area by dragging the split bar.

- **WHERE clause.** This text box allows you to insert an SQL WHERE clause to specify criteria that the data must meet before being selected.
- **Other clauses.** This text box allows you to insert a HAVING or an ORDER BY clause.

For more information about these clauses, see DataStage documentation.

## Using User-Defined Queries

Instead of using the SQL statement constructed by DataStage, you can enter your own SQL statement for each Sybase OC output link. To enter an SQL statement:

1. Select **User-defined SQL query** from the **Query type** list on the **General** tab of the **Output** page. The **User-defined** tab on the **SQL** tab is enabled.

You can edit the statements or drag and drop the selected columns into your user-defined SQL statement. You must ensure that the table definitions for the output link are correct and represent the columns that are expected. The result set generated from this statement returns at least one row.

If the property value begins with {FILE}, the remaining text is interpreted as a pathname, and the content of the file supplies the property value.

The SQL statement must generate a result set that matches the stage output column definitions.

If you specify multiple SQL statements, they are executed as one or more Transact-SQL command batches using a semicolon ( ; ) as the end-of-batch signal.

**Note:** If more than one result set is produced, only the first set is used.

2. Click **OK** to close this dialog box. Changes are saved when you save your job design.

## Sybase SQL Server Data Type Support

The following tables document the support for Sybase SQL Server data types. When creating DataStage table definitions for a Sybase table, set the SQL type, length, and scale attributes as noted.

### Character Data Types

The following table summarizes character data types for Sybase SQL Server, their DataStage SQL type definitions, and the corresponding length attributes that you need to set:

Sybase Data Type	DataStage SQL Type	Length	Notes
<i>char(n)</i>	Char	<i>n</i>	Sybase <i>char</i> values are blank padded to <i>n</i> characters.
<i>nchar(n)</i>	Char	<i>n * @@ncharsize</i>	Sybase <i>nchar</i> values are blank padded to <i>n</i> characters. <i>@@ncharsize</i> is a Sybase global variable that contains the byte length of a character in the Sybase server character set.
<i>varchar(n)</i>	VarChar	<i>n</i>	NA
<i>nvarchar(n)</i>	VarChar	<i>n * @@ncharsize</i>	<i>@@ncharsize</i> is a Sybase global variable that contains the byte length of a character in the Sybase server character set.
sysname	VarChar	30	The user data type supplied by Sybase, which is defined as <i>varchar(30)</i> .
<i>text</i>	Unsup-ported	NA	NA

## Numeric (Integer) Data Types

The following table summarizes the numeric (integer) data types for Sybase SQL Server and their DataStage SQL type definitions:

Sybase Data Type	DataStage SQL Type
<i>bit</i>	<i>Bit</i>
<i>tinyint</i>	<i>Tinyint</i>
<i>smallint</i>	<i>Smallint</i>
<i>int</i>	<i>Integer</i>

## Numeric (Decimal) Data Types

The following table summarizes the numeric (decimal) data types for Sybase SQL Server, their DataStage SQL type definitions, and the corresponding length and scale attributes that you need to set:

Sybase Data Type	DataStage SQL Type	Length	Scale	Notes
<i>decimal(p,s)</i>	Decimal	<i>p</i>	<i>s</i>	The full range of Sybase <i>decimal</i> values are supported without loss of precision.
<i>numeric(p,s)</i>	Numeric	<i>p</i>	<i>s</i>	The full range of Sybase <i>numeric</i> values are supported with no loss of precision.

## Numeric (Money) Data Type

The following table summarizes the numeric (money) data type for Sybase SQL Server, its DataStage SQL type definitions, and the corresponding length and scale attributes that you need to set:

Sybase Data Type	DataStage SQL Type	Length	Scale	Notes
<i>money</i>	Decimal	19	4	The full range of Sybase <i>money</i> values are supported with no loss of precision.

## Numeric (Approximate) Data Types

The following table summarizes the numeric (approximate) data types for Sybase SQL Server, their DataStage SQL type definitions, and the corresponding length attributes that you need to set:

Sybase Data Type	DataStage SQL Type	Length	Notes
<i>float(p)</i>	Float	<i>p</i>	DataStage Float values have a maximum precision of 15 digits. Some loss of precision will occur when reading data from Sybase <i>float(p)</i> columns where <i>p</i> is greater than 15.
<i>real</i>	Real	NA	NA
<i>double precision</i>	Double	NA	NA

## Date Data Types

The following table summarizes the date data types for Sybase SQL Server and their DataStage SQL type definitions:

Sybase Data Type	DataStage SQL Type	Notes
<i>datetime</i>	Timestamp Date <sup>d</sup> Time <sup>t</sup>	<p><sup>d</sup> The time component of a Sybase <i>datetime</i> value is lost when converted to a DataStage Date value. When writing a DataStage Date value to a Sybase <i>datetime</i>, the time component is set to midnight.</p> <p><sup>t</sup> The date component of a Sybase <i>datetime</i> value is lost when converted to a DataStage Time value. When writing a DataStage Time value to a Sybase <i>datetime</i>, the date component is set to the current date on the DataStage server machine.</p>
<i>smalldatetime</i>	Timestamp Date <sup>d</sup> Time <sup>t</sup>	<p><sup>d</sup> The time component of a Sybase <i>smalldatetime</i> value is lost when converted to a DataStage Date value. When writing a DataStage Date value to a Sybase <i>smalldatetime</i>, the time component is set to midnight.</p> <p><sup>t</sup> The date component of a Sybase <i>smalldatetime</i> value is lost when converted to a DataStage Time value. When writing a DataStage Time value to a Sybase <i>smalldatetime</i>, the date component is set to the current date on the DataStage server machine.</p>

## Binary Data Types

The following table summarizes the binary data types for Sybase SQL server and their DataStage SQL type definitions:

Sybase Data Type	DataStage SQL Type	Notes
<i>binary</i>	Unsupported	NA
<i>varbinary</i>	Unsupported	NA
<i>image</i>	Unsupported	NA
<i>timestamp</i>	Unsupported	The user data type supplied by Sybase, defined as <i>varbinary</i> (8). <i>timestamp</i> is not compatible with the DataStage Timestamp data type.

## Stored Procedure Support

You can call stored procedures from the server Sybase OC stage. The following restrictions apply:

- Specify input parameters as literal values. Passing row values as parameter values is not supported.
- Output parameters are not supported.
- You can call stored procedures as part of the BeforeSQL and AfterSQL statements. Any result sets generated by the procedure are discarded.
- You can also call stored procedures as part of the “User Defined SQL Statement” for output and reference links only. The stored procedure must generate a row result set that matches the stage output column definitions. Only one row result set is processed; any additional result sets are discarded.
- To call a stored procedure that is part of a group, precede the semicolon ( ; ) that separates the group name and procedure number with a backslash ( \ ). The backslash causes the stage to treat the semicolon as a regular character instead of an end-of-batch signal. For example, to call the `myprocgroup;2` stored procedure from the stage, use the following syntax:

```
execute myprocgroup\;2
```

- You cannot call stored procedures for input links. Passing row values as parameter values is not supported.
- The return code for the procedure is checked for Sybase errors (–1 to –99).
- Error numbers from –1 to –8 are treated as nonfatal warnings. Error numbers from –9 to –99 are treated as fatal errors. For more information on stored procedure return codes, see “Using Stored Procedures” in *Sybase Transact-SQL User’s Guide*.
- You must set the procedure’s execute mode to “chained” or “anymode”, as Sybase OC executes in chained transaction mode. To set execute mode, use the Sybase `sp_procxmode` procedure. For example:

```
sp_procxmode myproc, "anymode"
```

For more information on `sp_procxmode`, see “Transactions” in *Sybase Transact-SQL User’s Guide*.

