IBM InfoSphere DataStage and QualityStage
Version 9 Release 1

Connectivity Guide for ODBC

IBM
IBM InfoSphere DataStage and QualityStage
Version 9 Release 1

Connectivity Guide for ODBC

IBM
Note

Before using this information and the product that it supports, read the information in "Notices and trademarks" on page 137.
Chapter 1. Connector Migration Tool

To take advantage of the additional functionality that connectors offer, use the Connector Migration Tool to migrate jobs to use connectors instead of plug-in and operator stages.

The following table lists the stages that can be migrated to connectors and the corresponding connectors that they are migrated to:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Connector</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2Z stage</td>
<td>DB2 Connector</td>
</tr>
<tr>
<td>DB2 UDB API stage</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Load stage</td>
<td></td>
</tr>
<tr>
<td>DRS Stage</td>
<td>DRS Connector</td>
</tr>
<tr>
<td>Java Client stage</td>
<td>Java Integration Stage</td>
</tr>
<tr>
<td>Java Transformer stage</td>
<td></td>
</tr>
<tr>
<td>Netezza Enterprise stage</td>
<td>Netezza Connector</td>
</tr>
<tr>
<td>ODBC Enterprise stage</td>
<td>ODBC Connector</td>
</tr>
<tr>
<td>ODBC (Server) stage</td>
<td></td>
</tr>
<tr>
<td>SQL Server Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>Oracle OCI stage</td>
<td>Oracle Connector</td>
</tr>
<tr>
<td>Oracle OCI Load stage</td>
<td></td>
</tr>
<tr>
<td>Oracle Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>Teradata API stage</td>
<td>Teradata Connector</td>
</tr>
<tr>
<td>Teradata Enterprise stage</td>
<td></td>
</tr>
<tr>
<td>Teradata Load stage</td>
<td></td>
</tr>
<tr>
<td>Teradata Multiload stage</td>
<td></td>
</tr>
<tr>
<td>WebSphere® MQ stage</td>
<td>WebSphere MQ Connector</td>
</tr>
</tbody>
</table>

Migrating jobs to use connectors

To migrate jobs to use the connectors, you need to run the Connector Migration Tool.

To run the Connector Migration Tool, start it from the Microsoft Windows Programs menu or from the command line. If you start the tool from the command line, additional options that are not provided in the user interface are available.

The user interface leads you through the process of evaluating which jobs, shared containers, and stages to migrate. You select the jobs that you want to migrate, and beside each job name, the tool displays an icon that indicates whether or not the job can be fully migrated, partially migrated, or not migrated at all. To refine the list of jobs to evaluate, you can specify that only jobs that contain specific plug-in and operator stages be listed. The tool gives you a chance to make a backup of a job before you migrate it. You can make a backup copy of the job and then migrate the backup, or you can make a backup copy of the job and then migrate the original job. Either way, your original job is never lost. The job is migrated and
placed in the same folder as the original job, and the log file \texttt{CCMigration.log}, which records the results of the migration, is created in the current directory.

The Connector Migration Tool command line options provide the same functionality as the user interface, as well as a few additional options. Using the command line, you can perform these additional tasks:

- Specify a list of job names to be considered for migration.
- Specify a list of shared container names to be considered for migration.
- Specify a list of stage type names to limit the jobs that are considered for migration.
- Run a practice migration, where the actual migration does not take place but the possible results of the migration are placed in the log file. You can review the results and then refine the migration as necessary before you run the actual migration.
- Produce a report of jobs and their stages and stage types

Note:
- The Connector Migration Tool does not read environment variables at the operating system level. Environment variables are read only if they are defined within InfoSphere DataStage at the Project level or at the Job level. Project level environment variables are read first, then overwritten by Job environment variables. Environment variables with blank default values are ignored by the Connector Migration Tool. The default values of the environment variables are migrated, but the run-time values are not migrated.
- Throughout this documentation, the term "job" refers to parallel shared containers and server shared containers, as well as IBM® InfoSphere® DataStage® jobs.

Using the user interface to migrate jobs

Use the Connector Migration Tool to view which jobs and stages are eligible for migration and then migrate them to use connectors rather than plug-in and operator stages.

About this task

You use the same project connection details to connect to the Connector Migration Tool as you use to connect to the InfoSphere DataStage and QualityStage® Designer or InfoSphere DataStage and QualityStage Director Client. You must have sufficient user privileges to create and modify the jobs that you are migrating.

Procedure

1. Choose \texttt{Start} > \texttt{Programs} > \texttt{IBM InfoSphere Information Server} > \texttt{Connector Migration Tool}.

2. In the Log on window, complete these fields:
   a. In the \texttt{Host} field, enter the host name of the services tier. You can specify an optional port by separating it from the host name with a colon. The host name that you specify here is the same one that you specify when you start the Designer client, for example, mymachine:9080).
   b. In the \texttt{User name} field, enter your InfoSphere DataStage user name.
   c. In the \texttt{Password} field, enter your InfoSphere DataStage password.
   d. In the \texttt{Project} field, enter the name of the project. To access an InfoSphere DataStage server that is remote from the domain server, specify the project name.
name in full as server:[port]/project. As an alternative, you can press the button adjacent to the Project field to display a dialog box from which you can select the fully-qualified project name.

e. Click OK. An icon indicates the status of each job. A gray icon indicates that the job cannot be migrated. A gray icon with a question mark indicates that the job might be successfully migrated.

3. Display the jobs and stages to consider for migration:
   • Choose View > View all jobs to display all of the jobs in the project. This is the default view.
   • Choose View > View all migratable jobs to display all of the jobs that are in the project and that can be migrated to use connectors. Jobs that do not contain any stages that can be migrated are excluded from the job list.
   • Choose View > View jobs by stage types to open the Filter by stage type window.

4. Perform the following steps to analyze jobs:
   a. Highlight the job in the job list.
   b. Expand the job in the job list to view the stages in the job.
   c. Select one or more jobs, and click Analyze.

After analysis, the color of the job, stage, or property icon indicates whether or not it can be migrated. A green icon indicates that the job, stage, or property can be migrated. An red icon indicates that the job or stage cannot be migrated. An orange icon indicates that a job or stage can be partially migrated and that a property in a stage has no equivalent in a connector. A gray icon indicates that the job or stage is not eligible for migration.

Note: The Connector Migration Tool displays internal property names, rather than the names that the stages display. To view a table that contains the internal name and the corresponding display name for each property, from the IBM InfoSphere DataStage and QualityStage Designer client, open the Stage Types folder in the repository tree. Double-click the stage icon, and then click the Properties tab to view the stage properties.

5. Click Preferences and choose how to migrate the job:
   • Choose Clone and migrate cloned job to make a copy of the job and then migrate the copy. The original job remains intact.
   • Choose Back up job and migrate original job to make a copy of the job and then migrate the original job.
   • Choose Migrate original job to migrate the job without making a backup.

6. Select the jobs and stages to migrate, and then click Migrate.

The jobs and stages are migrated and are placed in the same folder as the original job. If logging is enabled, a log file that contains a report of the migration task is created. After a job is successfully migrated, a green checkmark displays beside the job name in the Jobs list to indicate that the job has been migrated.

Using the command line to migrate jobs

Run the Connector Migration Tool from the command line to use additional options that are not available in the user interface.
About this task

To run the Connector Migration Tool from the command line, you specify the command `CCMigration`, followed by a series of required and optional parameters. If the Connector Migration Tool is started from the command line, its user interface will be displayed if none of the options `-C`, `-M` or `-B` are specified. If any one of these options is specified, then the migration will proceed without any further interaction with the user. The command line options described below can therefore be used whether or not the user interface is displayed.

After a job is successfully migrated, a green checkmark displays beside the job name in the Jobs list to indicate that the job has been migrated.

Procedure

1. From the IBM InfoSphere DataStage client command line, go to the `<InformationServer>\Clients\CCMigrationTool` directory.
2. Enter the command `CCMigration`, followed by the following required parameters:
   - `-h host:port`, where `host:port` is the host name and port of the InfoSphere DataStage server. If you do not specify a port, the `port` is 9080 by default.
   - `-u user name`, where `user name` is the name of the InfoSphere DataStage user.
   - `-p password`, where `password` is the password of the InfoSphere DataStage user.
   - `-P project`, where `project` is the name of the project to connect to. To specify an InfoSphere DataStage server that is remote from the domain server, specify the fully qualified project name by using the format `server:[port]/project`.
   - One of the following:
     - `-M` If you specify this parameter, the original jobs are migrated, and backup jobs are not created.
     - `-B job name extension`, where `job name extension` is a set of alphanumeric characters and underscores. If you specify this parameter, the Connector Migration Tool creates backup jobs, names the backup jobs `source job name+job name extension`, and then migrates the original jobs. The backup jobs are saved in the same location in the repository as the source jobs.
     - `-C job name extension`, where `job name extension` is a set of alphanumeric characters and underscores. If you specify this parameter, the Connector Migration Tool clones the source jobs, names the cloned jobs `source job name+job name extension`, and then migrates the cloned jobs. The cloned jobs are saved in the same location in the repository as the source jobs.

   If you specify one of these options, the migration proceeds without requiring any additional user input. If you do not specify `-M`, `-B`, or `-C`, the user interface is displayed so that you can make additional choices for how to migrate the jobs.
3. Optional: Enter any of the following optional parameters:
   - `-L log file`, where `log file` is the file name and path for the log file that records the results of the migration.
   - `-S stage types`, where `stage types` is a comma-separated list of stage types. By default, the Connector Migration Tool migrates all stage types. Use this parameter to migrate only jobs that contain the specified stage types. If you specify both the `-S` and `-J` parameters, only the specified stage types within the specified jobs are migrated. If you specify the `-S` parameter and do not specify the `-C`, `-M` or `-B` parameter, only jobs that contain the specified stage
types appear in the job list that is displayed in the user interface. Limiting the jobs that are displayed can significantly reduce the startup time of the Connector Migration Tool.

- `-J job names`, where `job names` is a comma-separated list of jobs. By default, the Connector Migration Tool migrates all eligible jobs in the project. Use this parameter to migrate only specific jobs. If you specify the `-J` parameter and do not specify the `-C`, `-M` or `-B` parameter, only the specified jobs appear in the job list that is displayed in the user interface. Limiting the jobs that are displayed can significantly reduce the startup time of the Connector Migration Tool.

- `-c shared container names`, where `shared container names` is a comma-separated list of shared containers. By default, the Connector Migration Tool migrates all eligible shared containers in the project. Use this parameter to migrate only specific shared containers. If you specify the `-c` parameter and do not specify the `-C`, `-M`, or `-B` parameter, only the specified shared containers appear in the job list that displays in the user interface. Limiting the shared containers that display might significantly reduce the startup time of the Connector Migration Tool.

- `-R` If you specify this parameter, the Connector Migration Tool reports the details of the migration that would occur if the specified jobs were migrated, but does not perform an actual migration. The details are reported in the log file that is specified by using the `-L` parameter.

- `-a auth file`, where `auth file` is the file name that records the user name and password.

- `-A` If you specify this parameter, the Connector Migration Tool adds an annotation to the job design. The annotation describes the stages that were migrated, the job from which the stages were migrated, and the date of the migration.

- `-d job dump file`, where `job dump file` is the file name and path for a file where a list of jobs, shared containers, and stages is written. Using a job dump file is helpful when you want to determine which jobs are suitable for migration. You can use the `-d` parameter with the `-J`, `-c`, and `-S` parameters to list particular jobs, shared containers, and stage types, respectively.

- `-V` If you specify this parameter, the Connector Migration Tool specifies the target connector variant for migrated stages. The format of the list is a comma-separated list containing `{StageTypeName=Variant}`.

- `-v` If you specify this parameter with the `-d` command, the values of stage properties will be included in the report. If omitted, the report only contains stage names and types, but not the stage properties. This option is useful to identify jobs that have stages with certain property values. If this option is specified, then `-S` is ignored.

- `-T` If you specify this parameter, the Connector Migration Tool enables the variant migration mode. All connector stages found in jobs and containers whose stage type matches those listed by the `-V` command are modified.

- `-U` If you specify this parameter, the Connector Migration Tool enables the property upgrade migration mode. All connector stages found in jobs and containers whose properties match the conditions specified in the `StageUpgrade.xml` file are upgraded.
Example

The following command starts the Connector Migration Tool, connects to the project billsproject on the server dsserver as user billg, and migrates the jobs db2write and db2upsert:

```
CCMigration -h dsserver:9080 -u billg -p padd0ck -P billsproject -J db2write,db2upsert -M
```

Deprecated stages

Connectors, which offer better functionality and performance, replace some stages, which have been deprecated and removed from the palette. However, you can still use the deprecated stages in jobs, and add them back to the palette.

The following stage types have been removed from palette for the parallel job canvas:

- DB2Z
- DB2® UDB API
- DB2 UDB Load
- DRS
- Dynamic RDBMS
- Java Client
- Java Transformer
- Netezza Enterprise
- ODBC Enterprise
- Oracle 7 Load
- Oracle OCI Load
- Oracle Enterprise
- Teradata API
- Teradata Enterprise
- Teradata Load
- Teradata Multiload
- WebSphere MQ

The following stage type has been removed from the palette for the server job canvas:

- Dynamic RDBMS

When you create new jobs, consider using connectors instead of the deprecated stages. The following table describes which connector to use in place of which deprecated stages:

![Table 2. Stages and corresponding connectors](image)

<table>
<thead>
<tr>
<th>Deprecated stages</th>
<th>Connectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2Z</td>
<td>DB2 Connector</td>
</tr>
<tr>
<td>DB2 UDB API</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Enterprise</td>
<td></td>
</tr>
<tr>
<td>DB2 UDB Load</td>
<td></td>
</tr>
<tr>
<td>DRS</td>
<td>DRS connector</td>
</tr>
</tbody>
</table>
To use any of the deprecated stage types in new jobs, drag the stage type from the repository tree to the canvas or to the palette. From the repository tree, expand **Stage Types**. Under **Stage Types**, expand **Parallel** or **Server** depending on the stage that you want to use. Drag the stage type to the job canvas or to the palette.
Chapter 2. Configuring access to ODBC data sources

To configure access to ODBC data sources, you must install database client libraries and include the path to these installed libraries in the library path environment variable. For more information about setting environment variables, see the topic about setting environment variables.

Procedure
1. Install database client libraries.
2. “Configuring access to ODBC data sources.”

Configuring access to ODBC data sources

To use the ODBC data sources in a job, you must configure the ODBC driver and also the ODBC data source name (DSN) definitions.

Before you begin
• Install client libraries.
• Test the connection to the ODBC data source.
• On 64-bit Windows computers, make sure that you run the 32-bit version of the Microsoft ODBC Data Source Administrator C:\Windows\SysWOW64\odbcad32.exe, as InfoSphere Information Server is a 32-bit application. If you run the 64-bit version of the ODBC administrator application, the Netezza connector cannot to locate the specified data source name. If the ODBC administrator application is not accessible through the File menu by default, use the Windows Explorer to access the application.
• For more information about configuring ODBC connectivity in InfoSphere DataStage and configuring ODBC data source name definitions, see the IBM InfoSphere Information Server Planning, Installation, and Configuration Guide and .

Procedure
1. Configure the ODBC DSN definitions.

Table 3. Configuring the ODBC data source name definitions

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX or Linux</td>
<td>Set the ODBCINI environment variable to point to the .odbc.ini file. The .odbc.ini file contains the ODBC DSN definitions. Note: The ODBCINI environment variable is set in the dsenv script automatically as part of the InfoSphere Information Server installation process.</td>
</tr>
<tr>
<td>Microsoft Windows</td>
<td>The DSN definitions are managed by the ODBC driver manager application included with the operating system. The ODBC DSN definitions must be configured as System DSN definitions in the ODBC data source Administrator. The ODBCINI environment variable is not applicable on Microsoft Windows.</td>
</tr>
</tbody>
</table>
2. Add the path to the directory that contains the client libraries to the library path environment variable. The default path for client libraries is as follows:
   - On Windows, `C:\IBM\InformationServer\ODBCDrivers`. On the Microsoft Windows, the ODBC driver manager library is provided by the operating system. The location of the ODBC driver manager is automatically included in the PATH environment variable.
   - On Linux and UNIX, `/opt/IBM/InformationServer/Server/branded_odbc/lib`. The ODBC driver manager is included with InfoSphere Information Server.

3. On UNIX and Linux computers, to restart the server engine and the ASB Agent, enter the following command:

```
cd Install_directory/Server/DSEngine/bin
./uv -admin -stop
./uv -admin -start
```
```
cd Install_directory/ASBNode/bin
./NodeAgents_env_DS.sh
./NodeAgents.sh stopAgent
./NodeAgents.sh start
```

**Testing database connections by using the ISA Lite tool**

After you establish connection to the databases, test the database connection by running the IBM Support Assistant (ISA) Lite for InfoSphere Information Server tool.

For more information about the ISA Lite tool, see the topic about installation verification and troubleshooting.

**Setting the library path environment variable**

To apply an environment variable to all jobs in a project, define the environment variable in the InfoSphere DataStage and QualityStage Administrator. The values that are specified for the library path and path environment variables at the project or job level are appended to the existing system values for these variables.

**About this task**

For example, suppose that directory `/opt/branded_odbc/lib` is specified as the value for the library path environment variable at the project level. Directory `/opt/IBM/InformationServer/Server/branded_odbc/lib`, which contains the same libraries but in a different location is already in the library path that is defined at the operating system level or the `dsenv` script. In this case, the libraries from directory `/opt/IBM/InformationServer/Server/branded_odbc/lib` are loaded when the job runs because this directory appears before directory `/opt/branded_odbc/lib` in the values that are defined for the library path environment variable.

The name of the library path environment variable depends on your operating system.

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Library path environment variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Windows</td>
<td>PATH</td>
</tr>
<tr>
<td>HP-UX</td>
<td>SHLIB_PATH</td>
</tr>
<tr>
<td>IBM AIX®</td>
<td>LIBPATH</td>
</tr>
<tr>
<td>Other supported Linux and UNIX operating systems, and HP-IA</td>
<td>LD_LIBRARY_PATH</td>
</tr>
</tbody>
</table>
On Linux or UNIX operating systems, the environment variables can be specified in the dsenv script. InfoSphere Information Server installations on Windows operating system do not include the dsenv script.

**Setting the library path environment variable in the dsenv file**

On Linux or UNIX operating systems, you can specify the library path environment variables in the dsenv script. When environment variables are specified in the dsenv script, they apply to all InfoSphere DataStage projects that run under the InfoSphere Information Server engine.

**Before you begin**

Install the client libraries.

**Procedure**

1. Log in as the root user.
2. Back up the $ISHOME/Server/DSEngine/dsenv script. $ISHOME is the InfoSphere Information Server installation directory (/opt/IBM/InformationServer by default).
3. Open the dsenv script.
4. Add the path to the directory that contains the client libraries to the library path environment variable.
5. To stop and start the InfoSphere Information Server engine, enter the following commands:
   
   ```
   bin/uv -admin -stop
   bin/uv -admin -start
   ```
6. To change directory to the ASB Agent home directory, enter the following commands:

   ```
   cd Install_directory/ASBNode/bin
   ```
7. To stop and start the ASB Agent processes, enter the following commands:

   ```
   ./NodeAgents.sh stopAgent
   ./NodeAgents.sh start
   ```

**Results**

After you restart the ASB Agent process, the InfoSphere Information Server domain services (WebSphere Application Server) take approximately a minute to register the event.

**Setting the library path environment variable in Windows**

On the Windows operating system, both the library path and PATH environment variables are represented by the PATH. For InfoSphere Information Server engine and ASB Agent processes to detect changes in the environment variables, the changes must be made at the system level and the InfoSphere Information Server engine must be restarted.

**Before you begin**

Install the client libraries.

**Procedure**

1. To edit the PATH system environment variable, click Environment Variable in Advance System Settings, and then select PATH.
2. Click **Edit**, then specify the path to the directory containing the client libraries.
3. Click OK.
4. Restart the InfoSphere Information Server engine.
5. Restart the ASB Agent processes.
Chapter 3. ODBC sources

When you use IBM InfoSphere DataStage to access the ODBC application programming interface (API), you can choose from a collection of connectivity options. For most new jobs, use the ODBC connector, which offers better functionality and performance.

A connector is a component that provides data connectivity and metadata integration for external data sources, such as relational databases or messaging software. A connector typically includes a stage that is specific to the external data source.

The ODBC connector is one of several different stages that access external data sources in database management systems (DBMS). In addition to the ODBC Connector stage, the following stages are available:

- ODBC Enterprise stage, which is available only for parallel jobs
- ODBC stage, which is available only for server jobs

If you have jobs that use the older stages and want to use the connector, use the Connector Migration Tool to migrate jobs to use connector.

The ODBC connector provides some ease-of-use benefits to you:

- View data in a design-time environment.
- Create job parameters directly from the connector without having to define the parameters first.
- Save connection information that you specify in the stage as a data connection object, which means that you can reuse the object.
- Configure rules for reconciling data types between the source and target schemas to avoid runtime errors.
- Receive immediate feedback about any invalid values for properties on the Properties page.
- Reset a property value to its default value.
- View error messages that are generated when your jobs run.

The following table lists the scenarios where you might want to use one of the stages other than the ODBC Connector stage.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Stage to use instead of the ODBC Connector stage</th>
</tr>
</thead>
</table>
| Use stored procedures. | • Stored Procedure (preferred)  
• ODBC |
| In a lookup operation, return more than one result on the reference link for each input row. | ODBC (server jobs) |
Chapter 4. Prerequisites for the ODBC connector

The ODBC connector has no special installation requirements. However, there are configuration prerequisites that are specific to the ODBC connector.

Before you can use the ODBC connector in a job, you need to configure database drivers, driver managers, and data source names.

Database drivers

You must install and configure database drivers and at least one driver manager before you can use the ODBC connector.

Supported drivers are included with the installation of this product. For information about the supported drivers and how to configure them, refer to the IBM InfoSphere Information Server Planning, Installation, and Configuration Guide.

The following drivers have limitations or special configuration requirements when you use them with the ODBC connector:

**IBM Text File Driver**
- Supported data types. Only Numeric, Date, and VarChar are supported.
- Unsupported runtime data types. The following runtime data types are not supported:
  - Bit
  - Binary
  - LongNVarChar
  - LongVarBinary
  - LongVarChar
  - NChar
  - NVarChar
  - Time
  - Timestamp

**SQL Server driver**
- Large objects (LOBs) by reference. You cannot pass them by reference with this driver. You must transfer them inline.

**SQL Server native driver**
- Transfer large objects (LOBs) by reference. You cannot pass them by reference with this driver. You must transfer them inline.

**SQL Server wire driver**
- Transfer large objects (LOBs) by reference. You cannot pass them by reference with this driver. You must transfer them inline.
- View design-time data. To view design-time data that contains spaces, you must select the quoted identifiers check box on the Advanced tab of the driver setup window.
Supported data sources

A data source is a repository of data that can contain relational databases, XML files, and table-structured files.

To be a supported data source, the ODBC connector must be able to perform read, write, and lookup SQL statements and to exchange data between external data sources and the IBM InfoSphere DataStage data sets.

For information about the supported data sources, see IBM InfoSphere Information Server Planning, Installation, and Configuration Guide.

Data source prerequisites

Use the ODBC connector to connect to relational database management systems (DBMSs), files, and any other ODBC-supported data sources.

Before the connector can open a table or file to read and write data, a connection must be defined to the data source that contains that table or file.

For the ODBC connector, the ODBC driver manager establishes the connection. To establish your data sources correctly, the following requirements must be met:

- The driver and driver manager must be installed on the same local system where the connector is installed and running. The data source can be on a remote system, and you can also connect to multiple data sources.
- For parallel jobs, the driver and driver manager must be installed on every node where the connector runs.

You can also connect to only one ODBC driver manager at a time.

When you configure the driver manager and you are logged into the Designer client, you can see a list of data source names in the Data source property by clicking the Data source button. This button is available only when you click inside the property. If you are working in a design environment that is not connected to the server, you can type a value in the Data source property.

Data source names

You must define names for each data source that are specific to the operating system, driver manager, and driver.

For information about how to configure ODBC data source names, see IBM InfoSphere Information Server Planning, Installation, and Configuration Guide.
Chapter 5. ODBC connector

You can use the ODBC connector in your jobs to read data, write data, look up data, and filter data.

ODBC connector jobs

You can use the ODBC connector in your jobs to read data, write data, look up data, and filter data.

The role of the connector is determined by the links that are attached to it in the job and how the connector is configured. The following links can be used with the ODBC connector:

Input links
The connector receives data.

Output links
The connector sends data.

Reference links
The connector participates in normal or sparse lookups.

Reject links
The connector filters rows of data that meet your specified criteria and passes the rows that meet this criteria to another stage.

You can use one input link or one output link or one reference link. Additionally, you can use a reject link with an input link.

Example link configurations

Here are some example link configurations for the ODBC connector:

* Input link to ODBC connector
* Output link to ODBC connector
* Input link to ODBC connector to reject link
* Output link to ODBC connector to reject link

Job design and the ODBC connector

You can use the ODBC connector in your jobs to perform different types of tasks, such as testing your connection to the data source, performing lookup operations, defining rejected record criteria, and as a source or a target of data.

After you add the connector to your job, you can specify links and other stages to define the role of the connector in the job.

Job execution order and its effect on job design

At run time, the ODBC connector uses the properties that you specify at design time to define how the Data Definition Language (DDL) statements and Data Manipulation Language (DML) statements are executed and how errors are handled. The ODBC connector reads the property information in the following order:

1. Schema reconciliation
You determine the levels of schema reconciliation that the connector performs between your design-time schema and your external schema by using the properties in the **Schema reconciliation** property group and any other associated properties.

2. **Table action**
   
   You can specify any of the following operations:
   
   • Create a table.
   • Replace a table.
   • Truncate a table.
   • Append to a table.
   
   You can also specify the DDL statement for the action, or you can specify that the DDL statement is generated automatically.

3. **Before SQL**
   
   If you set the **Before/After SQL** property to **Yes** and specify an SQL statement in the **Before SQL** property, the connector runs the statement. If you specify multiple statements in the **Before SQL** property, the connector runs each additional statement in sequence.

4. **Before SQL (node)**
   
   If you set the **Before/After SQL** property to **Yes** and specify an SQL statement in the **Before SQL (node)** property, the connector runs the statement on each processor node.

5. **Write mode**
   
   The connector performs all of the data manipulation that you specify in the **Write mode** property. The connector runs all of the SQL statements that are related to the **Write mode** property value. For example, if you set the **Write mode** property to **Insert then Update**, the connector runs the SQL statement that you specify in the **Insert statement** property and then the SQL statement that you specify in the **Update statement** property.

6. **After SQL (node)**
   
   If you set the **Before/After SQL** property to **Yes** and specify an SQL statement in the **After SQL (node)** property, the connector runs the statement on each processor node.

7. **After SQL**
   
   If you set the **Before/After SQL** property to **Yes** and specify an SQL statement in the **After SQL** property, the connector runs the statement. If you specify multiple statements in the **After SQL** property, the connector runs each additional statement in sequence.

**Note:** There are certain data types that the ODBC connector cannot generate a DDL for at design time or at run time. When designing a job, you are warned if you define such a data type. At run time, the connector inserts the nearest matching data type and does not issue a warning. The workaround is to specify the DDL yourself rather than have the connector do it automatically.

**Configuring the connector as a source**

To configure the connector as a source, you must define the connection to a Teradata database, ODBC data source, WebSphere MQ queue manager, IBM DB2 database, specify the properties for the output link, and define columns for the data that the connector must read.
About this task

In the source context, the connector extracts or reads data from an external Teradata databaseODBC data sourceWebSphere MQ queue managerIBM DB2 database.

When the WebSphere MQ connector receives a new message from a queue, certain information about the structure of the message can be determined. For example, the connector can determine whether the message contains only text or whether it contains format headers in the data.

Restriction: Only one connection is valid from a single thread. If your transaction involves SEBridge running both the WebSphere MQ plug-in and WebSphere MQ connector on the same thread, a warning message at MQCLOSE and a unrecoverable error at MQCMIT occurs alerting you that the connection is shared and transactions are overlapped between the two MQ stages. To work around these errors, you can add an interprocess stage between the transformer and MQCC to run the plug-in and connector separately.

Procedure

1. In the job canvas, add the connector to the job.
2. Add the stage that follows the connector in the job flow.
3. Add the output link from the connector to the next stage. Right-click the connector and then drag it on the next stage. The connector now has an output link that connects it to the next stage in the job flow.
4. Double-click the connector to open the stage editor.
5. On the Properties tab, define the connection properties for the Teradata databaseODBC data sourceWebSphere MQ queue managerDB2 database.
6. Optional: On the Advanced tab, specify the custom processing settings.
7. Specify information about the output link:
   a. On the Output tab, select the output link.
   b. On the Properties tab, define the usage properties for the link.
   c. On the Columns tab, define the column metadata for the link.
   d. Optional: On the Advanced tab, you can specify custom buffering settings for the link.
8. Click OK to save your changes and to close the stage editor.

Configuring the connector as a target

To configure the connector as a target, you must define the connection to a Teradata databaseODBC data sourceWebSphere MQ queue managerIBM DB2 database, specify the properties for the input link, and define columns for the data that the connector will write.

About this task

In the target context, the connector connects to the external Teradata databaseODBC data sourceWebSphere MQ queue managerIBM DB2 database and inserts, updates, or deletes data.

When the WebSphere MQ connector sends a message to the queue, the connector uses the schemas in the job. The connector reads the data from the payload column and uses the data as the payload for the outgoing message.
Procedure

1. In the job canvas, add the connector to the job.
2. Add the input link from the previous stage in the job flow to the connector. Right-click the previous stage and then drag it on the connector. The connector now has an input link that connects it to the previous stage in the job flow.
3. Double-click the connector to open the stage editor.
4. On the Properties tab, define the connection properties for the Teradata database ODBC data source WebSphere MQ queue manager DB2 database.
5. Optional: On the Advanced tab, specify custom processing settings.
6. Specify information about the input link:
   a. On the Input tab, select the input link.
   b. On the Properties tab, define the usage properties for the link.
   c. On the Columns tab, define the column metadata for the link.
   d. Optional: On the Advanced tab, you can specify custom buffering settings for the link.
   e. Optional: On the Partitioning tab, you can specify custom partitioning settings for the link.
7. If the connector has a reject link, specify how to send data to this link:
   a. On the Reject tab, select the reject link, then define the reject conditions for the link.
   b. Optional: On the Advanced tab, specify custom buffering settings for the link.
8. Click OK to save your changes and to close the stage editor.

Record ordering:

If the connector has multiple input links, you can control the processing order of input data across links.

Use one of these methods to control the processing order of input data:

Specifying the order of input data by input link:

When the connector uses multiple input links, you can control the sequence in which records are processed by ordering the links.

About this task

The order in which you specify the links on the Link Ordering tab determines the order in which the records in the links are processed for each unit of work.

Procedure

1. From the stage editor, select an input link.
2. Click the Link Ordering tab.
3. Click a link that you want to reorder, and use the arrow buttons to move the link up or down.

Specifying the order for records:

If a connector has multiple input links, you can control the order of record processing by specifying the order for the records.
Procedure
1. Double-click the connector stage icon to open the connector properties.
2. Set Record ordering to one of the following:
   - **All records** specifies that all records are processed from each link in order.
   - **First record** specifies that one record is processed from each link, in turn, until all records from all links have been processed.
   - **Ordered** specifies that records are processed from each link in an order you specify by using the **Key column**, **Null order**, and **Case-sensitive** properties.
3. If you choose **Ordered**, complete these additional properties:
   a. **Key column** – Specify the name of the column to use as the sort key.
   b. **Sort order** – Specify **Ascending** or **Descending**.
   c. **Null order** – Specify where to sort null values in the sort order. The choices are **Before** or **After**.
   d. **Case sensitive** – Specify whether or not text comparisons are case-sensitive. The choices are **Yes** or **No**.

**Rejecting records that contain errors**
When the connector includes a reject link, records that meet specified reject criteria are automatically routed to the target stage of the reject link, and processing continues for the remaining records.

**Before you begin**
- Create a job that includes the connector and required links.
- Define a connection to the database.
- Set up column definitions on the links.
- Specify the write mode and the target table.

**About this task**
When you configure a reject link, you select one or more conditions that control when to reject a record and send it to the target stage that receives the rejected records. You can also choose to include the error code and error message that is generated when a record fails. If you do not define a reject link or if you define a reject link but a failed record does not match any of the specified reject criteria, the connector reports a Fatal error and stops the job.

If the connector has multiple input links, you can specify multiple reject links. You use the **Reject from link** field to specify the input link to associate with the reject link.

**Procedure**
1. Configure a target stage to receive the rejected records.
2. Right-click the connector and drag to create a link from the connector to the target stage.
3. If the link is the first link for the connector, right-click the link and choose **Convert to reject**. If the connector already has an input link, the new link automatically displays as a reject link.
4. Double-click the connector to open the stage editor.
5. Click the **Reject** tab.
6. If the connector has multiple reject links, in the **Reject from link** field, select the input link to associate with the reject link.
7. In the **Reject rows based on selected conditions** list, select one or more conditions to use to reject records.

**Note:** If you do not choose any conditions, none of the rows are rejected. In this case, any error that occurs while the records are being written to the target table results in job failure.

8. Use one of the following methods to specify when to stop a job because of too many rejected rows:
   - In the **Abort when** field, select **Percent**. Then in the **Abort when (%)** field, enter the percentage of rejected rows that will cause the job to stop. In the **Start count after (rows)** field, specify the number of input rows to process before calculating the percentage of rejected rows.
   - In the **Abort when** field, select **Rows**. Then in the **Abort after (rows)** field, specify the maximum number of reject rows allowed before the job stops.

9. In the **Add to reject row** list, select additional columns to include in the rejected data. For example, if you are using the Oracle connector, you might select the **ERRORCODE** and **ERRORMESSAGE** columns, which contain information about why a row is rejected.

### Lookup operations with the ODBC connector

The ODBC connector with the Lookup stage can perform a join operation between one or more external data source tables and an IBM InfoSphere DataStage input resource.

An example of an input resource is a data set or a sequential file. The output result of the join is an InfoSphere DataStage data set.

With normal lookup and sparse lookup jobs, the ODBC connector is connected to a Lookup stage. In both lookup types, the key column values determine how the data from two different sources is joined together. These two lookup types differ in the location in which the rows of data are processed. You must specify a WHERE clause and the specific columns that are used in a sparse lookup. Both lookup types look the same on the job canvas.

#### Creating a lookup operation with a connector:

You can use the ODBC TeradataDB2 connector to create a lookup operation.

**Before you begin**

You must create a job first.

**About this task**

After you create the lookup job, how you configure it determines whether it is a normal lookup or a sparse lookup.

**Procedure**

1. Add an input resource stage such as a Data set stage or a Sequential File stage to the job canvas.
2. Add a Lookup stage as the output stage to the input resource stage.
3. Add an output resource stage, such as a Data set stage or a Sequential File stage, after the Lookup stage. This stage serves as an output stage to the Lookup Stage.
4. Add the ODBCTeradataDB2 connector directly above the Lookup stage.
5. Connect the connector to the Lookup stage with a reference link.
6. Save the job.

Example

This example shows the result of this procedure.

What to do next

Now, you are ready to configure this job to perform a normal lookup operation or a sparse lookup operation.

Configuring the connector for normal lookup operations:

You can use the ODBCTeradataDB2 connector to perform a normal lookup operation. In this lookup operation, the connector retrieves all of the records and allows the Lookup stage to process the records.

Before you begin

You must create a lookup operation job first. You also must define your columns in the input stage for the Lookup stage and the output stage for the Lookup stage. For information about how to define the columns in these stages, refer to the stage-specific documentation.

Procedure
1. Double-click the connector on the job canvas to open the stage editor.
2. On the Output tab, select the reference link.
3. Configure the properties on the Properties tab.
   a. Set Lookup type to normal.
   b. Define and test your connection properties in the Connection section.
c. Type your SELECT statement in the **Select statement** property by using the following format: `select * from table_name`. Note that this syntax means that many records are retrieved as opposed to the records that are retrieved in a sparse lookup operation.

d. Optional: Configure any other properties on the **Properties** tab.

4. Define your columns on the **Columns** tab of the connector stage editor.
   a. In the stage editor, click the **Columns** tab.
   b. Define the columns that you want to use from the database to which the connector is connected.

5. Click **OK** to save the changes to the connector.

6. Double-click the Lookup stage to open the stage editor.
   a. Map the required columns from your data input link to the output link. You can drag them or copy and paste them.
   b. Map the required columns from your lookup table to the output link.
   c. Specify the key column or columns for the lookup. Drag or copy the key column from the data link to the **Key Expression** field in the lookup table link. Only key columns can be used for key expressions.
   d. Define any conditions for a lookup failure by clicking the **Constraint** icon in the menu.
   e. Select the appropriate value for the **Lookup Failure** column and click **OK**. If you select **Reject**, you must have a reject link and target stage in your job configuration to capture these records.

7. In the Lookup Stage stage editor, click **OK**.

8. Save, compile, and run the job.

**What to do next**

For more information about this topic, refer to the Parallel Job Developer's Guide

**Configuring the connector for sparse lookup operations:**

You can use the ODBCTeradataDB2 connector to perform a sparse lookup operation.

**Before you begin**

You must create a lookup operation job first. You must define your columns in the input stage for the Lookup stage and the output stage for the Lookup stage. For information about how to define the columns in these stages, refer to the stage-specific documentation.

**About this task**

In this lookup operation, the connector receives the records from the input stage, and then the connector performs the lookup operation directly on the external resource. The connector then generates the output records.

You can use the sparse lookup method only in parallel jobs.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor.
2. On the Output tab, select the reference link.
3. Configure the properties on the Properties tab.
   a. Set Lookup type to sparse.
   b. Define and test your connection properties in the Connection section.
   c. In the Select statement property, in the select part of the SELECT statement, the asterisk wildcard (*) does not work in a sparse lookup. Therefore, specify every column in the database and delimit the columns by commas. You must specify all columns, even if you do not use them in this lookup. The following syntax is an example of the first part of the SELECT statement: select Field001,Field002,Field003.
   d. After your table name, specify a WHERE clause. Key columns that follow the WHERE clause must have the word ORCHESTRATE and a period added to the beginning of the column name. ORCHESTRATE can be capitalized or lowercase letters. An example of this is the following column name: ORCHESTRATE.Field001. The following SELECT statement is an example of proper syntax: select Field001,Field002,Field003 from MY_TABLE where ORCHESTRATE.Field001 = Field001.
   e. Optional: Configure any other properties on the Properties tab.

4. Define your columns on the Columns tab of the connector stage editor.
   a. In the stage editor, click the Columns tab.
   b. Define the columns that you want to use from the database to which the connector is connected.

5. Click OK to save the changes to the connector.

6. Double-click the Lookup stage to open the stage editor.
   a. Map the required columns from your data input link to the output link. You can drag them or copy and paste them.
   b. Map the required columns from your lookup table to the output link.
   c. Define any conditions for a lookup failure by clicking the Constraint icon in the menu.
   d. Select the appropriate value for the Lookup Failure column and click OK. If you select Reject, you must have a reject link and target stage in your job configuration to capture these records.

7. In the Lookup Stage stage editor, click OK.

8. Save, compile, and run the job.

What to do next

For more information about this topic, see IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide.

Changing the lookup operation type in the connector:

You can change from a normal lookup to a sparse lookup or from a sparse lookup to a normal lookup. You must edit your SELECT statement in the Select statement property.

Before you begin

You must create a job first where the connector is defined in a lookup operation.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. On the Output tab, select the reference link.
3. Change the value of the **Lookup type** field from normal to sparse or from sparse to normal.

4. Make edits to the SQL statement in the **Select statement** property:
   - To change from a normal lookup operation to a sparse lookup operation:
     a. Optional. You can replace the wildcard asterisk (*) for the column names in your normal lookup SELECT statement by specifying each column name that is delimited by commas.
        For example, the `SELECT *` syntax for a normal lookup now becomes `SELECT Field001, Field002, Field003` for a sparse lookup.
     b. Add a `WHERE` clause to your `SELECT` statement and then specify the key field column from the source stage. You must add the word **ORCHESTRATE**, followed by a period (.), in front of any field that resides in an external resource. You can type the **ORCHESTRATE** in uppercase or lowercase letters.
        An example of this is the following column name: **ORCHESTRATE.Field001**.
     c. Establish the relationship between the key field column in the source stage with the database table field. Type the following information: a space after the key field name, an equal sign (=), another space, and then the database table field name.
        The following example is a sparse lookup statement:
        ```sql
        SELECT Field001, Field002, Field003 FROM MY_TABLE WHERE
        ORCHESTRATE.Field001 = Field001
        ```
        In this example, **Field001** is not preceded by the word **ORCHESTRATE** because it is a field in **MY_TABLE**.
   - To change from a sparse lookup operation to a normal lookup operation:
     a. Double-click the connector to open the stage editor.
     b. On the **Output** tab, select the reference link.
     c. Change **Lookup type** from sparse to normal.
     d. Optional. You can replace the columns at the beginning of the **SELECT** statement with a wildcard asterisk (*).
     e. Delete everything in the statement beginning with the word **WHERE** to the end of the statement.

5. Make any other changes in the connector stage editor that you need.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>No column changes</td>
<td>Click OK to save your changes. If you are configuring a normal lookup, skip to step 7.</td>
</tr>
<tr>
<td>Column changes</td>
<td>Go to the next step.</td>
</tr>
</tbody>
</table>

6. Optional: Click the **Columns** tab and make your changes. Click **OK** to save your changes.

7. Required: For normal lookup operations only, open the Lookup stage. In the left side of the stage editor for the Lookup stage, draw a connection between the primary key fields on the left side.

**Configuring the connector to connect to the data source**

Use the properties in the **Connection** section on the **Properties** tab to configure the connection of the data source for the connector.
Before you begin

You must create a job and add the connector to it first.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. Configure the values for the properties in the Connection section on the Properties tab.

What to do next

Now, you can test the connection.

Testing the connector connection to the data source

You can test the connection to your data source in the stage editor before you compile and run your job.

Before you begin

You must configure the properties in the Connection section of the stage editor first.

Procedure

1. In the Connection section, click Test.

<table>
<thead>
<tr>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The connection values are correct.</td>
<td>The connection is made and a confirmation message is displayed. You have completed this procedure.</td>
</tr>
<tr>
<td>The connection values are incorrect.</td>
<td>An error message is displayed. Go to the next step.</td>
</tr>
</tbody>
</table>

2. Edit the connection property values.
3. Repeat steps 1 and 2.

What to do next

Now, you can save this connection information, along with additional connector information, for reuse.

Information reuse and the ODBC connector

The ODBC connector provides ways for you to save and reuse information such as connection specifications and property values in a data connection object, metadata, and job parameters.

The following topics describe how to work with these information pieces:

- Data connection objects
  - “Saving connection information as data connection objects” on page 28
- Metadata
  - “Saving metadata in the connector” on page 32
  - “Selecting a job parameter in a connector property” on page 29
Saving connection information as data connection objects

You can save your connection information as a data connection object. The object can then be reused by other stages.

Before you begin

You must configure the connection to the data sourcequeue managerdatabase in the Connection section of the stage editor.

Procedure

1. In the Connection section, click Save. The Data Connection window is displayed, including the existing connection values.
2. Complete the remaining fields in this window, including the name of the data connection object, a description of the object, and the folder in which to save this object.
3. To save the object, click OK. The following text is added to the Connection section to specify the newly saved object:
   (Associated data connection: connection_object_name)

Reusing data connection objects in your connector job

Data connection objects that are stored in the repository can be loaded into a stage definition. You can create a data connection object once and then reuse the object in your job design whenever you need access to the same external resource.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. In the Connection section, click Load. A list of data connection objects for this connector is displayed.
3. Select the data connection object and click OK. All of the settings in the data connection object are inserted into the stage editor. The following text is added to the Connection section to specify the selected object:
   (Associated data connection: connection_object_name)

Creating job parameters in the connector properties

You can create new job parameters in a connector property without having to define them first. Job parameters allow you to define flexible, reusable jobs.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. Click a property value for which you want to create a job parameter. If you can create a parameter for that property, you see the job parameter button.
3. Click the button and select New Parameter.
4. Specify the parameter that you want to add.
   a. Accept the default value for the Parameter name field or type a new value.
   b. Accept the default value for the Prompt field or type a new value.
   c. Select the value for the Type field from the list.
   d. In the Default Value field, type the value that you want the job to use at run time. If you want to be prompted at run time for this value, leave this value blank. If you select Encrypted as the Type, a separate window is displayed. You must type the encrypted value, then type it again for confirmation, and then click OK.
e. Optional: In the **HelpText** field, type any description.

f. Click **OK**.

**Results**

For more information about this topic, see IBM InfoSphere DataStage and QualityStage Designer Client Guide.

**Example**

You can use job parameters in jobs where a database connector connects to a database. In this example, you want to run the same job against different data sources. You define job parameters so that the connector can dynamically connect to the data source that you specify at runtime. At design-time, you define job parameters for the **Data source**, **Username**, and **Password** properties. When the job runs, you are prompted with a default value if you specified one or a blank value. You can change the default value or type a new value. The connector then uses this value to connect to the data source.

**Selecting a job parameter in a connector property**

You can select a job parameter in any connector property that supports job parameters.

**Before you begin**

Job parameters must be defined for your connector type.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor.
2. Click a property value for which you want to insert a job parameter. If you can use a parameter for that property, you see the job parameter button.
3. Click the button and select the parameter from the list.

**Results**

The job parameter is inserted into the property value. For more information about this topic, see IBM InfoSphere DataStage and QualityStage Designer Client Guide.

**Removing a job parameter from a connector property**

To remove a job parameter from a connector property, you must use the job parameter button.

**About this task**

This button is available only when you click inside the property.

**Procedure**

1. Click the property value that you want to change.
2. Select **Clear Parameter**.
Results

If there is a default value for that property, the default value is now displayed. Otherwise, type or select the value that you want for this property.

Metadata and the ODBC connector

When the connector has access to the runtime server in a design-time environment, you can work with metadata at the column level or at the table level.

You can import metadata from your local repository or a remote repository if the remote repository is configured to be shared in the administrator application. You can also save metadata for later reuse by the same type of connector.

Importing metadata by columns

You can import metadata by columns on the Columns tab of the connector stage editor.

Before you begin

The metadata must be saved in the repository first.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. Click the Columns tab.
3. Click Load.
4. Select the table from the tree that contains the metadata and click OK.
5. Use the right arrow (>) button to move the columns that you want to load from the Available columns list to the Selected columns list. If you want to move all of the columns at once, click the double right arrow (>>). To move columns back to the Available columns list from the Selected columns list, use the left arrow (<) for one column or the double left arrow (<<) for all columns.
6. Click OK. The selected column metadata is loaded into the Columns tab of the stage editor.

Importing metadata for the ODBC connector by tables

You can import metadata by tables from an ODBC data source and store the data as a table definition in the repository.

About this task

You can import table definitions to your project or make them available to other projects if you configure your project to share metadata in the administrator client. You access the Import Connector Metadata wizard to import the tables.

Procedure

1. Use any of the following ways to access the Import Connector Metadata wizard:

<table>
<thead>
<tr>
<th>Method</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer client</td>
<td>From the Import menu, select Table Definitions &gt; Start Connector Import Wizard.</td>
</tr>
</tbody>
</table>
2. When the wizard is displayed, your next step depends on whether you defined a database for importing metadata yet. Also, you might not see certain windows in the wizard, depending on the way that you accessed the wizard. Follow the pages in the wizard to complete the metadata import.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>You are a first-time user. You have not defined a location and a database for importing metadata.</td>
<td>Go to the next step.</td>
</tr>
<tr>
<td>You have already defined a location or database.</td>
<td>Skip to step 7.</td>
</tr>
</tbody>
</table>

3. In the Data source location page of the Import Connector Metadata wizard, click **New location**, select the system that hosts the database from **Host name where database resides**.
4. In the Shared Metadata Management window, select the location in the repository tree and then click **Add new database**.
5. In the Add new database window, type values for **Name**, **Server**, **Location**, **Vendor**, **Version**, **Short Description**, and **Long Description** as needed and click **OK**.
6. Click **Close**.
7. Select the database in **Database name** and click **Next**.
8. Select the name of the connector type from the list and click **Next**.
9. If you have DataConnection objects defined in your project that you want to use, click the **Load** link.
10. Navigate to the folder for existing data connection objects, select the object, and click **Open**. The **Database**, **Username**, and **Password** fields are populated with the corresponding data from the data connection object.
11. To ensure that you can connect to the database, click the **Test connection** link. If the test is not successful, check the connection field values. Then click **Next**.
12. Type or select the schema name in the **Schema** list. Decide whether you want to include tables and views and type the name of a search filter that you want to use when you import the metadata. Then click **Next**.
13. Select the tables and views that you want to import. You can also include primary key and foreign key information and indices.
14. Optional: Click the **View data** link to look at the actual data in the table that matches your selection criteria. Click **Close** to close the data window and to return to the wizard. Then click **Next**.
15. Review the import details, and then click **Import**.
16. Select the folder to which you want to import the metadata or table definition and click **OK**.

**Results**

The metadata is imported and is displayed in the selected folder with a unique table definition icon. This icon uniquely identifies that the table definition is imported into the shared area of the repository. The table definition is also now available to other projects and to other components.

For more information about this topic, see IBM InfoSphere DataStage and QualityStage Designer Client Guide.

**Saving metadata in the connector**

You can save your metadata to reuse it in another connector. Metadata in this context is the schema that you define on the **Columns** tab.

**Before you begin**

You must define your schema on the **Columns** tab.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor.
2. Click the **Columns** tab.
3. Click **Save**.
4. Type new or change existing field values in this window and click **OK**.
5. Complete the save procedure.
   a. Optional: Change the name of the table definition in **Item name**.
   b. Use the tree to navigate to an existing folder or create a new folder in which to save the table definition.
   c. Double-click the folder. The project path is displayed in the **Folder path** field.
   d. Click **Save**.

**Results**

The table definition, including the metadata, is now available for reuse.

**What to do next**

For more information about this procedure, see IBM InfoSphere DataStage and QualityStage Designer Client Guide.

**Design-time data and the connector**

With the ODBC TeradataWebSphere MQDB2 connector, you can view the data in your Teradata databaseODBC data sourceWebSphere MQ applicationIBM DB2 database at design time.

When you view data at design time, you can see whether the following things are true:

- Your connection configuration isconnection settings for the local queue manager or remote queue manager are accurate
• Your SQL statement is accurate and that you can see the columns that you want
to see; settings for the local queue or remote queue are accurate and that you can
see the messages that you expect the job to process at runtime
Although you read the message data on a remote or local queue, you do not
destroy that data in the process of reading it.
• The data in your Teradata database; ODBC data source; DB2 database is what you
expect and whether it is valid

Viewing data at design time
You can view the data in your Teradata database; ODBC data source; WebSphere MQ
application; IBM DB2 database.

Procedure
1. Double-click the connector on the job canvas to open the stage editor.
2. Configure the properties in the Connection section.
3. To review queue message data on a remote queue, you must select Client for
the Mode property value.
4. Test the connection settings by clicking Test.
   If the connection test fails, troubleshoot the connection settings. Until you can
   access the Teradata database; ODBC data source; WebSphere MQ queue
   manager; DB2 database, you cannot view data.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector with an output link</td>
<td>Define your SQL statement in Select statement.</td>
</tr>
<tr>
<td>Connector with an input link</td>
<td>Define your Insert statement, Update statement, or Delete statement.</td>
</tr>
</tbody>
</table>

5. Type or select the name of the queue for which you want to view data.
6. Click View data.

Data manipulation at run time for the connector
You can control how the connector manipulates data at run time with different
SQL statements.

When the connector is a source or is in a lookup operation, you can create a
SELECT statement to read data from a data source in the Select statement
property.

When the connector is a target, you can select from a list of different SQL
statements and statement combinations in the Write mode property:

Insert  Use the Insert statement property to define the INSERT statement that
inserts data into the data source at run time.

Update  Use the Update statement property to define the UPDATE statement that
updates the data in the data source at run time.

Delete  Use the Delete statement property to define the DELETE statement that
deletes data from the data source at run time.

Insert then update Use the Insert statement property and Update statement property to
define the INSERT statement and the UPDATE statement. Update statements are run on only the rows or records that did not get inserted into the database.

**Update then insert**

Use the Update statement property and Insert statement property to define the UPDATE statement and the INSERT statement. Insert statements are run on only the rows or records that did not get updated in the database.

**Delete then insert**

Use the Delete statement property and Insert statement property to define the DELETE statement and the INSERT statement. For each input record, the connector first tries to delete the matching rows in the target table and then runs the insert statement to insert the record as a new row in the target table. The connector runs the insert statement regardless of whether rows were deleted or not.

**Large objects (LOB) in the connector**

You can define how LOBs are transferred by the connector in these methods: inline and by reference.

**Note:** Reading data from tables containing LOB columns is only supported where the LOB columns are the last columns listed in the select statement. Similarly, viewing data from tables containing LOB columns is only supported where the LOB columns are the last columns defined in the table.

The following table describes the types of LOBs and the IBM InfoSphere DataStage data type for them:

<table>
<thead>
<tr>
<th>LOB</th>
<th>Description</th>
<th>InfoSphere DataStage data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary LOB (BLOB)</td>
<td>Represents a LOB in binary form</td>
<td>LongVarBinary</td>
</tr>
<tr>
<td>Character LOB (CLOB)</td>
<td>Represents a LOB that is held in character form. The database defines the character set.</td>
<td>LongVarChar</td>
</tr>
<tr>
<td>NLS Character LOB (NCLOB)</td>
<td>Represents a LOB that is held in the national language set (NLS) for the database. Used for multibyte character sets.</td>
<td>LongNVarChar</td>
</tr>
</tbody>
</table>

**LOB transfers inline**

When you use the inline method, the actual data is transferred between the connector and the database driver.

If you decide to transfer your LOBs in this way, you must set **Enable LOBs by reference** to No and set **Array size** to 1.
LOB transfers by reference

When you use the by reference method, a locator that represents the LOB is transferred between the connector and the database driver, instead of the actual data. The actual data is then read later on in the job by another connector that uses the inline method.

Transferring large objects (LOBs) by reference

You can configure a connector to transfer LOBs by reference. Because a connector is the only stage type that can do this, the stage that transfers the LOB inline later in the job must also be a connector.

Before you begin

You must create a job in which the connector is the source first. You also must have LOBs in the data that you want to transfer.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. On the Columns tab, define or import your table definition. At least one of the columns must be a LOB and there must be at least one primary key field.
3. Configure the properties on the Properties tab for transferring the LOBs by reference.
   a. Set Enable LOB references to Yes.
   b. In the Columns property, you can type the column name, or you can use the Available columns button to provide a list of LOB columns from which to select. This button is available only when you click inside the property. If you click the button, select the column or columns in the Select Columns window and click OK.
4. Click OK to save your connector settings.

Schema reconciliation for the connector

The schemas for job runs are design-time schemas that are defined in the job definition and external schemas that are on the external resource.

A schema defines a structure and the type of contents that each data element within the structure can contain.

Based on the schema reconciliation rules in the ODBC connector, the connector can determine how to resolve or how not to resolve differences between these schemas. These rules help to avoid the following results that can occur when there is a mismatch between the fields in the schemas:

- Runtime errors
- Extra conversions

Schema reconciliation rules for the connector

The connector uses rules to reconcile the differences between the design-time schema and the external schema.

You can configure some of these rules in property settings on the Properties tab. The connector uses the following rules:
Presence of fields
The connector attempts to match the design-time schema field names with
the external resource field names that the connector accesses.

When you use the connector as a source of data, and there are more fields
in the design-time schema than in the external schema in parallel jobs,
these extra fields are dropped, and informational messages are generated.
Conversely, when there are more fields in the external schema than in the
design-time schema, these extra fields are added to the schema, and
informational messages are generated.

When you use the connector as a target of data and there are more
columns in the external schema than in the design-time schema in parallel
jobs, these fields are added to the schema, and informational messages are
generated. If you have more fields defined in the design-time schema than
you have defined in the external schema, you can determine whether these
extra fields in the design-time schema are ignored in the Drop unmatched
columns property. If you set Drop unmatched columns to Yes, the extra fields
are ignored and informational messages are generated. Otherwise, an
unrecoverable error message is generated, and the job is terminated.

When you use the connector as part of a lookup operation, the unmatched
columns are always dropped.

Data types and sizes of fields
The data type reconciliation of fields occurs after the field-matching level.
The data types of the design-time schema fields and the external schema
fields that are matched in the previous level are compared. If the data
types in both schemas are the same, no conversion is required and nothing
needs to be reconciled.

If the data types are different, reconciliation is attempted. However, some
data types cannot be converted. For example, if you have a string data
type in your design-time schema in a target context and the field in the
external schema is an integer (for example, int8), the conversion is not
reliable. The string can have nonnumeric characters. Therefore, this
conversion is not allowed. An error message is generated, and the job is
terminated. However, if you had the opposite situation where you had an
integer in the design-time schema and a string in the external schema, this
conversion can be made.

If you set Fail on type mismatch to Yes and the schemas cannot be
reconciled to maintain data integrity, an unrecoverable error message is
generated and the job is terminated. Otherwise, a warning message is
generated.

Attributes of fields
Field attribute reconciliation occurs after the fields match and the data
types are converted. The attributes of the design-time schema fields and
the external schema fields are compared, and the connector attempts to
reconcile any attribute differences. Some attributes include minimum and
maximum length (size), precision, and character encoding. You want to
prevent conversions that include data write operations from a large field to
a smaller field or from a less restrictive field into a more restrictive field.

For the field size, you can set Fail on size mismatch to Yes. If you set
Fail on size mismatch to Yes and issues such as large fields getting
written to smaller fields or less restrictive fields into more restrictive fields
occur during schema reconciliation, an unrecoverable error message is generated and the job is terminated. Otherwise, a warning message is generated.

**Configuring schema reconciliation rules for the connector**

You can configure some of the rules that the ODBC connector uses to attempt to resolve differences between design schemas and external schemas.

**Procedure**

1. In the stage editor, click either Input tab or the Output tab, then select either the input link or the output link.
2. Click **Schema reconciliation** to open the group of sub-properties.
3. Specify whether the job fails if you have a size mismatch in one or more of your fields during schema reconciliation in the **Fail on size mismatch** property.
4. Specify whether the job fails if you have type mismatch that cannot be reconciled in one or more of your fields in the **Fail on type mismatch** property.
5. For input links only, use the **Drop unmatched fields** property to determine how to process fields that do not exist in the input database schema.

**Parallel execution and partitioned data for the ODBC connector**

For parallel jobs, the connector supports parallel reads.

You set **Enable partitioning** to Yes to allow parallel execution. At run time, when you specify that the data is not partitioned, the connector uses an unmodified WHERE clause to return all of the rows of data. When you specify that the data is partitioned, the connector modifies the WHERE clause to return a subset of the total number of rows. The connector performs the WHERE clause modification on each query that is on each processing node. Each node then receives a unique subset of rows. The subsets of rows are determined by the partitioning method that you specify in the **Partitioning method** property.

**Partitioning data for parallel reads for the connector**

You must configure the ODBC connector to perform parallel reads. A parallel read is when the data is broken up into subsets of data, and then the data is concurrently read by different processing nodes.

**Before you begin**

You must create a job where the connector is defined as a source first. Before you run a job for parallel reads, you must configure your processing nodes in addition to performing the steps in this procedure.

**About this task**

You can perform parallel reads for source jobs and normal lookup jobs.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor.
2. On the Output tab, select the reference link for normal lookups only or the output link that you want to configure for parallel reads.
3. Set **Enable partitioning** to Yes.
4. In the **Partitioning method** property, select the partitioning method that you want to divide your data into subsets.

5. In the **Column name** property, type the name of the key field column in which the data is partitioned when the job is run.

**Results**

When this job is run, the WHERE clause in your SELECT statement is modified to return a subset of rows that are read by each processing node.

**What to do next**

For more information about partition configuration and logical nodes, refer to the Parallel Job Developer’s Guide

**Specifying transaction isolation levels for the connector**

You can specify how transactions or units of work are processed when other concurrently running transactions run. The degree of isolation that you specify for a transaction can also include commitment information.

**Before you begin**

Before you can specify the isolation level, you must create a job where you define the connector in a source context, a target context, or in a lookup operation.

**About this task**

Transactions are committed in the target stage when the end of wave marker is detected by the target stage.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor.
2. Select the link that you want to configure.
3. Set **Isolation level** to the value that you want.

**SQL statements for the connector**

You can use SQL statements in the ODBCDB2 connector to work with data in your Data Manipulation Language (DML) statements and to work with tables in your data definition language (DDL) statements.

You can use the ODBC connector with the following kinds of SQL statements:

<table>
<thead>
<tr>
<th>Type of statement</th>
<th>Processing level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>Job</td>
<td>Specify when an SQL statement is run. This action can be either before or after the job and either once per job or once per processing node per job.</td>
</tr>
<tr>
<td>Data Manipulation Language (DML)</td>
<td>Row</td>
<td>Specify SQL statements that manipulate data at the row level, such as INSERT and UPDATE.</td>
</tr>
</tbody>
</table>
Table 6. Types of SQL statements for the ODBC connector (continued)

<table>
<thead>
<tr>
<th>Type of statement</th>
<th>Processing level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Definition Language (DDL)</td>
<td>Table</td>
<td>You can specify SQL statements that manipulate data at the table level, such as CREATE TABLE and DROP TABLE. You can also specify TRUNCATE TABLE statements.</td>
</tr>
</tbody>
</table>

**Data Definition Language statements for tables**

The connector uses Data Definition Language (DDL) statements to create new tables and to replace, append, or truncate existing tables.

You work with the **Table action** property and sub-properties to define your DDL statements.

You can create your DDL statement in these ways:

- Type your statement at design-time in the **Create statement**, **Drop Statement**, or **Truncate statement** properties, depending on the action.
- Automatically generate a generic SQL statement at run time with the **Generate** button. The generated DDL is based on the columns that you define on the **Columns** tab. You might have to edit this statement to make it specific to your situation.

**Generating Data Definition Language (DDL) statements in the connector at design time:**

You can specify the connector to automatically generate sample, generic DDL statements for your CREATE TABLE, DROP TABLE, or TRUNCATE TABLE statements at design time.

**Before you begin**

The connector generates a generic version of the DDL statement. You need to edit this statement, especially if you have data types that are not supported for DDL generation in your database.

**About this task**

You must create a job where the connector is defined as a target first.

**Procedure**

1. Double-click the connector on the job canvas to open the stage editor.
2. In the **Table action** property, select the type of DDL statement that you want to generate.
   - **Append**
     - The data for the columns that you specify on the **Columns** tab are added to the destination table.
   - **Create, Replace, or Truncate**
     - The group of properties for that table action are enabled.
3. Set the appropriate generate statement at runtime property to **No**.
• Generate create statement at runtime
• Generate drop statement at runtime
• Generate truncate statement at runtime

4. In the Fail on error property, specify whether the job fails if the DDL statement cannot be run successfully.

5. Optional: Configure other properties on the Properties tab.

6. Click the Columns tab.

7. Make sure that you define the column names and their data types correctly.

8. Click the Properties tab.

9. Generate a generic DDL statement that you can edit, or type your own statement. To generate a baseline DDL statement, click in the statement property value, whether it is Create statement, Drop statement, or Truncate statement, and click Generate. This button is available only when you click inside the property.

10. Edit the statement as needed.

11. Click OK to save the job.

Generating Data Definition Language (DDL) statements in the connector at runtime:

The connector can automatically generate the DDL statements for your CREATE TABLE, DROP TABLE, or TRUNCATE TABLE statements at run time.

Before you begin

When you generate the DDL statement at run time, the connector queries the database driver to determine the native data type that corresponds to each data field. Because data types vary from database to database, the connector cannot generate the DDL statement for all data types. Binary and LongVarBinary InfoSphere DataStage data types are not supported DDL statement generation at run time. When you specify the following data types on the Columns tab, the connector replaces the data type in the first column with the data type in the second column to generate the DDL statement at run time.

<table>
<thead>
<tr>
<th>Design-time data type selected on the Columns tab</th>
<th>Data type used by the connector to generate the DDL statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>LongNVarChar</td>
<td>VarChar</td>
</tr>
<tr>
<td>LongVarChar</td>
<td>VarChar</td>
</tr>
<tr>
<td>NChar</td>
<td>Char</td>
</tr>
<tr>
<td>NVarChar</td>
<td>VarChar</td>
</tr>
</tbody>
</table>

If you have any of these data types in your database, you must edit the generated DDL statement.

About this task

You must create a job with the connector as a target first.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. In the **Table name** property, select the type of DDL statement that you want to generate:

   **Append**
   The data for the columns that you specify on the Columns tab are added to the destination table.

   **Create, Replace or Truncate**
   The group of properties for that table action are enabled.

3. Set the appropriate generate statement at runtime property to **Yes**:
   - **Generate create statement at runtime**
   - **Generate drop statement at runtime**
   - **Generate truncate statement at runtime**

4. Specify whether you want the job to fail if the DDL statement cannot be run successfully in the **Fail on error** property.

5. Optional: Configure other properties on the **Properties** tab.

6. Click the **Columns** tab.

7. Make sure that the columns do not contain any of the data types listed above. If you do have any of these data types, you must change them. Otherwise, the job fails.

8. Click **OK** to save the job.

**Data Manipulation Language statements for data manipulation**

The connector uses Data Manipulation Language (DML) statements to manipulate data in data sources.

For the source context (output links), the SELECT statements select data from a data source during a **READ** operation.

For the target context (input links) the INSERT, UPDATE, DELETE, INSERT then UPDATE, UPDATE then INSERT, and DELETE then INSERT statements work with the specified data in a data source during a **WRITE** operation. The multiple-action statements, such as INSERT then UPDATE, specify the order in which these statements are run.

These statement properties are all sub-properties of the **SQL** property on the **Properties** tab.

You can create your DML SQL statement in several different ways:

- Type your statement in the property value.
- Open a small statement window to more easily type a multiple-line statement.

   - Click in the property value and then press ![more button] to open this window.

- Use the SQL builder to graphically build your statement. Press **Build** at the end of the statement value to select the version of SQL builder that you want to use and to open the SQL builder in that version.

- Automatically generate a SQL statement at run time.

When you use either of the first two methods, make sure that you specify an association between each table column and each statement parameter in your SQL statement. If you do not do this, only limited schema reconciliation is performed. The following example is a statement where the association between columns and statement parameters has not been made:
INSERT INTO Table VALUES(ORCHESTRATE.A, ORCHESTRATE.B)

The table columns in Table were not identified. The following example is a correct statement that includes the associations with columns X and Y:
INSERT INTO Table(X,Y) VALUES(ORCHESTRATE.A, ORCHESTRATE.B)

Generating SQL statements in the connector at run time:

You can configure the connector to automatically generate the SELECT, INSERT, UPDATE, DELETE, INSERT then UPDATE, UPDATE then INSERT, or DELETE then INSERT statement at run time.

Before you begin

You must create a job first. The connector can be used in a source context or a target context.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. In the navigator, click the output link or the input link, depending on the job that you create.
3. Set Generate SQL to Yes.
4. In the Table name property, type the name of the table for the SQL statement.
5. For target contexts (input links) only, in the Write mode property, select the type of statement that you want to generate.
6. Optional: Configure other properties on the Properties tab.
7. Click the Columns tab.
8. Define the columns that you want to use in the SQL statement.
9. Click OK to save the job.

Generating SQL statements in the connector at design time

You can configure the connector to generate SQL statements at design time in their statement properties.

Before you begin

Create a job that includes a connector as a source or target.

About this task

You can generate the SQL statement text only for those statement properties that have the Generate SQL statement option in the Build list.

Note: Under some circumstances, the connector requires a connection to generate SQL statements. When a user name and password are not supplied and a connection is required, a connection is made by using the user who is running the ASB Agent service.

Procedure

1. Double-click the connector on the job canvas to open the stage editor.
2. In the navigator, click the output or input link, depending on the type of job that you create.
3. Set Generate SQL at runtime to No.
4. In the **Table name** property, type the name of the table for the SQL statement.
5. For jobs in target context (input links), select the type of statement you want to generate in the **Write mode** property.
6. On the **Columns** page, define the columns to use in the SQL statement.
7. Click the **Properties** tab.
8. Click the **Build** button that is associated with the statement property, and select **Generate SQL statement** from the list.

   **Note:** The **Generate SQL statement** option will only be available for statements which that connector supports generating at design time. In some cases a connector may only support generating the SQL at runtime during job execution.

9. Click **OK** to save the job.

**Validating SQL statements in the connector at design time**
After you generate or write a SQL statement, you can validate the statement during job design.

**About this task**
You can validate the SQL statement text only for those statement properties that have the **Validate SQL** option in the Build list.

**Note:** Under some circumstances, the connector requires a connection to validate SQL statements. When a user name and password are not supplied and a connection is required, a connection is made by using the user who is running the ASB Agent service.

**Procedure**
1. Save the job.
2. Click the **Build** button that is associated with the statement property, and select **Validate SQL**. The **Validate SQL** option is enabled only if the statement property contains a value and this option will only be available for statements which the target RDBMS supports validating.

**Results**
The connector validates the SQL statement by preparing the statement with the RDBMS it supports. If the SQL contains error, an error message is shown.

**ODBC connector properties**
The ODBC connector properties define how the connector operates in a job. There are different properties that are available depending upon the context in which you use the connector: source context, target context, or request context for lookup operations.

All of the properties that are displayed in either the **Connection** section or the **Usage** section of the **Properties** tab are listed in alphabetical order.

**After SQL**
Use this property to specify the SQL statement that is run once per job after any parallel processing occurs.
This property is available only if you set Before/After SQL to Yes.

You can type more than one statement in this property. Separate each statement with a semicolon (;). For server jobs, if you enable the Fail on error property, the job stops if an error occurs. For parallel jobs, the job does not stop if an error occurs.

To specify that this SQL statement is run once on each node or logical processor that can be a computer or a partition on a computer, use the After SQL (node) property instead of this property.

Click [ ] to open a statement window in which to type your SQL statement. This button is available only when you click inside the property.

Valid values are determined by the specific database.

**After SQL (node)**

Use this property to specify the SQL statement that is run once per node or logical processor that can be a computer or a partition on a computer after all data is processed.

This property is available only if you set Before/After SQL to Yes.

You can type more than one statement in this property. Separate each statement with a semicolon (;). Use the Fail on error property to specify whether the job stops when an error occurs.

To specify this SQL statement to be run once for the entire job after all parallel processing completes, use the After SQL property instead of this property.

Click [ ] to open a statement window in which to type your SQL statement. This button is available only when you click inside the property.

Valid values are determined by the specific database.

**Array size**

Use this property to specify the number of records or rows that are used at one time for all read and write database operations. The setting in this property can affect performance.

In lookup scenarios, the Array size value applies to output rows or records only. The input records or rows are processed one record or row at a time. If your data includes large objects (LOBs), you must set Array size to 1. Otherwise, a warning is issued and the job fails.

The Record count property value must be a multiple of the value of the Array size property.

Valid values are from 1 up to a database-specific maximum integer. The default value is 2000.
**Autocommit mode**

Use this property to specify when the transactions are committed and whether the connector or the database driver commits the transactions.

If you set *Autocommit mode* to On, the following statements are true:
- The driver determines when the transactions are committed. The connector is passive in this setting.
- The driver commits the transactions after every SQL statement is processed.

If you set *Autocommit mode* to Off, the following statements are true:
- The connector commits the transactions.
- The transactions are committed at every multiple of the value specified in *Record count*.

The default value is Off.

**Before/After SQL**

Use this property to determine whether you can access the sub-properties to define SQL statements that are run before or after data processing occurs.

If you set *Before/After SQL* to Yes, the sub-properties are available for you to use. If you set *Before/After SQL* to No, the sub-properties are not available to you. The default value is No.

**Before SQL**

Use this property to specify the SQL statement that is parsed and run once per job before any processing occurs.

This property is available only if you set *Before/After SQL* to Yes.

If you set *Table action* to Append, you cannot specify a CREATE TABLE statement in this property because the destination table must already exist.

You can type more than one statement in this property. Separate each statement with a semicolon (;). Use the *Fail on error* property to specify whether the job stops when an error occurs.

To specify that this SQL statement is run once on each node or logical processor, use the *Before SQL (node)* property instead of this property.

Click to open a statement window in which to type your SQL statement. This button is available only when you click inside the property.

You must include a CREATE INDEX statement as part of this statement if all of the following statements are true:
- You are using an IBM DB2 database on the mainframe.
- Your table contains primary keys or unique columns.

If you do not include the CREATE INDEX statement, a table is created if set *Table action* to Create or Replace. However, if you try to insert data into that table, an error message is generated and the job fails.
Valid values for **Before SQL** are determined by the specific database.

**Before SQL (node)**

Use this property to specify the SQL statement that is run once per node or logical processor before any data is processed.

This property is available only if you set **Before/After SQL** to Yes.

You can type more than one statement in this property. Separate each statement with a semicolon (;). Use the **Fail on error** property to specify whether the job stops when an error occurs.

To specify that this SQL statement is run once for the entire job before all parallel processing completes, use the **Before SQL** property instead of this property.

Click ![...](image) to open a statement window in which to type your SQL statement. This button is available only when you click inside the property.

Valid values are determined by the specific database.

**Case sensitive**

Use this property to specify whether or not text comparisons are case-sensitive.

If the connector uses multiple input links and you choose **Ordered** in the **Record ordering** field, use the **Case sensitive** field to specify whether or not text comparisons in a sort operation are case-sensitive.

**Code page**

Select the code page that the ODBC connector uses to communicate with the database driver.

The following values are available:

- **Default**
  
  The connector expects the data to be in the code page of the engine operating system where the connector runs. If a different code page is used, the connector converts the data. This value is the default.

- **Unicode**
  
  The data and property values, such as the user credentials in the **Connection** section and the SQL statements, are passed to the database driver in Unicode.

  If you have tables with more than one code page, you must select this option. An example of this is an Oracle database table with a VARCHAR2 column and an NVARCHAR column. Each of these columns has its own character set defined for it. To read the data in these columns correctly, you must select **Unicode** as the code page.

- **User-specified**
  
  When you select this option, the **Code page name** property is displayed. You must specify the name of the code page in the **Code page name** property. The name must comply with the Internet Assigned Numbers Authority (IANA) standards, such as UTF-8 or ISO-8859-1. The ODBC connector converts the data to the specified code page.
**Code page name**

Use this property to specify the name of the code page that is compatible with the data source.

This property is required if you set Code page to User-specified. The name must comply with the Internet Assigned Numbers Authority (IANA) standards, such as UTF-8 or ISO-8859-1. Valid values are up to 256 characters.

**Column delimiter**

Use this property to specify the delimiter used between column values in the log.

The Column delimiter property is available when you specify Yes for the Log column values on first row error property. The Column delimiter property is not available when Write mode property is set to Bulk load.

The supported values for the Column delimiter property are:
- Space - The default column delimiter is space.
- Newline - The column delimiter is a new line.
- Tab - The column delimiter is tab.
- Comma - The column delimiter is comma.

**Column name**

Use this property to specify the name of the column on which data is partitioned.

The minimum values and maximum values in the Column name property define a range that is added to the WHERE clause on each player node or logical processor to specify the partitions for that node.

Valid values are up to 256 characters.

**Example**

In this example, you have an integer column named userID and all of the records in this table have a value of between 0 and 100 for userID. There are four nodes in this example. The range for the first node is a minimum value of 0 and a maximum value of 24; the second node range is 25 and 49, and so on. So the first node receives all of the records with a userID value between 0 and 24.

**Columns**

Use this property to select the key column or key columns that contain large objects (LOBs) that are passed by using locator or reference information.

If the schema contains LOB columns that you want to pass inline as opposed to by reference, do not specify those columns in this property. If you are not passing LOBs inline, you must also set Array size to 1.

Click Available columns to select from the list of available LOB columns. You can select more than one column. This button is available only when you click inside the property.
Create statement

Use this property to specify the SQL statement to create a new database table.

This property is available only if you set the following property values:

- **Write mode** to Insert
- **Table action** to Create or Replace
- **Generate create statement at runtime** to No

To create a table in a Sybase database, you must set the Data definition language in transaction option in the Sybase database.

When you set **Table action** to Create and a table already exists with the same name as the one that you want to create, the job step that contains the ODBC connector stops with an error if you set **Fail on error** to Yes. If you set **Fail on error** to No, the database error is generated, and a warning message is issued.

When you set **Table action** to Replace, no error is generated. The schema of the Designer client data set determines the schema of the new table. The table is created with simple default properties. To create a complex create table statement (for example, a table with partitions), you must manually type the SQL statement.

Click **Generate** to automatically generate a generic SQL statement. This button is available only when you click inside the property. The generated statement is not database-specific and you might need to modify it, especially because of unique data types in different databases. You can manually modify the statement, or you can delete the generated statement and type your own statement.

Valid values are determined by the specific database.

Data source

Use this property to specify the data source that is used for all connections.

A data source is a repository of data that can contain a relational database, XML files, table-structured files, or other objects.

Click **Data source** to select from a list of available data sources. This button is available only when you click inside the property.

Valid values are a minimum of one character up to either the maximum for the specific database driver or 256 characters.

Delete statement

Use this property to specify the SQL statement to delete specified rows from an existing database table.

This property is available only if you set the following property values:

- **Write mode** to Delete or Delete then insert
- **Generate SQL** to No

The following is an example of a DELETE statement where table1 is the name of the table. You must precede the ID of every target table name with the word ORCHESTRATE that is followed by a period. ORCHESTRATE can be capitalized or typed in lowercase letters.
delete from table1 where ID=ORCHESTRATE.ID

Valid values are determined by the specific database.

Drop statement

Use this property to specify the SQL statement to drop an existing database table.

This property is available only if you set the following property values:

- **Write mode** to Insert
- **Table action** to Replace
- **Generate drop statement at runtime** to No

If the table that you want to drop does not exist and you set **Fail on error** to No, an error is generated. However, the job continues processing. If you set **Fail on error** to Yes, the job stops.

Click **Generate** to automatically generate a generic SQL statement. You can manually modify the generated statement, or you can delete the generated statement and type your own statement.

Valid values are determined by the specific database.

Drop unmatched fields

Use this property to specify whether fields that do not exist in the input database schema are dropped as part of schema reconciliation.

The default value is Yes, which means that the connector ignores any fields in the design-time schema that do not match fields in the database schema. If you set **Drop unmatched fields** to No, the connector attempts to reconcile the schemas according to its schema reconciliation rules. If there are unmatched fields in the design-time schema, the schema is not changed and the connector causes the job to fail.

Enable LOB references

Use the properties in this group to specify whether large object (LOB) columns in the **Columns** property are passed through the connector by using locator information. If the schema contains LOB columns and you set **Enable LOB references** to No, all LOB columns are passed inline. If you set **Enable LOB references** to Yes, only the specified columns are passed by reference. The remaining LOB columns are passed inline.

If you set **Enable LOB references** to Yes, the value that is passed as a locator is read from the database when the target stage runs. But, the rest of the data is read when the source stage runs. You also must set **Array size** to 1 if **Enable LOB references** is Yes.

The following IBM InfoSphere DataStage data types are treated as LOBs:
- LongVarBinary for database binary large object (BLOB) columns
- LongVarChar for database character large object (CLOB) columns
- LongNVarChar for database national character large object (NCLOB) columns

The default value is No.
**Enable partitioning**

Use this property to specify whether parallel reads are allowed.

This property is available only in the source context.

If you set **Enable partitioning** to Yes, the WHERE clause in the SQL statement is modified during processing to return a subset of the number of rows that would otherwise be returned if an unmodified query is used. This query modification occurs on each node or logical processor so that each node receives a different subset of rows to read. Use the **Partitioning method** property to determine how to partition the rows.

The default value is No that specifies that sequential reads are performed instead of parallel reads.

**Enable quoted identifiers**

Use this property to specify whether quotation marks are used to enclose all object names in your Data Definition Language (DDL) and Data Manipulation Language (DML) statements.

The default value is No. Depending on the database to which you are connecting, if quotation marks are not used, the object names can be converted to uppercase letters. This conversion can cause a problem when the job that contains the statement runs.

The following DDL statement is an example of the generated statement when you set **Enable quoted identifiers** to Yes:

```
create table "myTable"("myInt" number(20), "myString" varchar(50))
```

Certain database drivers require additional settings to use quoted identifiers.

For more information about connection values for specific databases and database drivers, see *DataDirect Connect DataDirect Connect XE for ODBC User’s Guide and Reference* (odbcref.pdf).

**End of data**

Use this property to specify whether the end of wave marker is inserted for the last remaining set of records. This scenario is invoked when the number of records in that last set is less than the value in the **Record count** property.

This property is available only if you set **End of wave** to a value other than None.

Units of work are transactional units. The end of wave marker indicates the end of a unit of work or transaction. The transaction unit is committed in a source stage after the end of wave marker for the transaction unit is successfully passed through the job. If you set this property to No and you have fewer records than specified in the **Record count** property, the following actions occur:

1. This last, smaller set of records is read by the connector.
2. The transaction is committed on the source stage.
3. These records are read by the next stage in the job. Also, these records cannot be rolled back to the source stage because the transaction was already committed on that stage.
4. If you set **Message read mode** to Delete (under transaction), after the records are read and are committed, they are removed from the queue.

The default value is Yes.

**End of wave**

Specify how the end of wave markers are handled in all of the transactions.

Units of work are transactional units. The end of wave marker indicates the end of a unit of work or transaction. The transaction unit is committed in a source stage after the end of wave marker for the transaction unit is successfully passed through the job. When the connector is used in a source context (that is, with an output link), the connector defines the unit of work by generating an end of wave marker as defined by the **End of wave** property.

The default value is None.

The following values are available:

- **None**  No end of wave marker is inserted into the data set.
- **Before**  End of wave markers are inserted before the transactions are committed.
- **After**  End of wave markers are inserted after the transactions are committed.

**Fail on error**

Use this property to specify whether the job is terminated when a database error occurs.

The default value is Yes.

If you set **Fail on error** to No and a database error occurs, but the SQL statement completes successfully, the following actions occur:

1. A warning message is generated.
2. The job continues to run.

If you set **Fail on error** to Yes and a database error occurs, a fatal error is generated, and the job is terminated.

**Fail on row error**

Use this property to specify if the connector should log a fatal error message and stop the job when an error occurs while writing a record to the database.

If you set this property to No and a record could not be written to the database, the connector logs a warning message and continues processing the remaining input records.

If you set this property to Yes and a record could not be written to the database, the connector logs an error message and the job stops.

The default value for the property depends on the type of job in which the ODBC connector stage is running:

- For parallel jobs, the default value is Yes.
- For server jobs, the default value is No.
Note that if a reject link is defined for the stage, this property is not available and automatically defaults to Yes.

If the input link to the ODBC connector stage is coming from a Transformer stage that has been configured to reject the rows that the ODBC connector stage could not write to the database, the **Fail on row error** property must be set to No in order to allow the Transformer stage to route those rows to the reject link.

**Fail on size mismatch**

Use this property to specify how any data size differences are handled that occur when the schema is validated at run time.

Size limitations are enforced when your design-time metadata is validated against the table definition in the database. This validation does not impact the actual reading or writing of the data. The connector attempts to read and process the data that it accesses as smoothly as possible.

If you set **Fail on size mismatch** to Yes and the connector detects possible incompatibilities when it compares the metadata during schema reconciliation, the job is terminated with an error message. The error message specifies the incompatible schema fields.

If you set **Fail on size mismatch** to No, and the sizes of the fields do not match, a warning message is generated. This message specifies the incompatible field sizes and that data truncation can occur. However, the job continues to run.

With both values, if the field sizes are compatible in both schemas, no message is generated and the job continues to run.

The default value is Yes.

**Fail on type mismatch**

Use this property to specify how to handle data type differences that occur when the schema is validated at run time.

Data types are compared when your design-time metadata is validated against the table definition in the database. The connector attempts to read and process the data that it accesses without errors as it attempts to reconcile the differences.

If you set **Fail on type mismatch** to Yes and the connector detects possible incompatibilities when it compares the metadata during schema reconciliation, the job is terminated with an error message. This error message specifies the incompatible schema fields.

If you set **Fail on type mismatch** to No, and the data types of the fields cannot be converted, a warning message is generated. This message specifies the incompatible data types and that data corruption can occur. However, the job continues to run.

Some data types can be reconciled; others cannot be reconciled. For example, if you have a string data type that is mapped into an integer data type, this data might be reconciled if the string does not contain any alpha characters. However, if you have a date data type that is mapped into a Boolean data type, reconciliation is attempted. But, the data cannot be reconciled, and the job is terminated.
Generate create statement at runtime
Use the properties in this group to determine whether you manually create or automatically generate a CREATE TABLE statement at run time.

The Generate create statement at runtime property is available only if you set the following property values:

- **Write mode** to Insert
- **Table action** to Create or Replace

The default value is No.

Generate drop statement at runtime
Use the properties in this group to determine whether you manually create or automatically generate a DROP TABLE statement at run time.

The Generate drop statement at runtime property is available only if you set the following property values:

- **Write mode** to Insert
- **Table action** to Replace

The default value is No.

Generate SQL
Use this property to specify whether the connector generates the SELECT, INSERT, UPDATE, or DELETE statement at run time.

If you set Generate SQL to Yes, the SQL statements are automatically generated at run time. You cannot specify them at design-time. The statements are generated by using the table name that you specify in **Table name** and the information that you specify on the **Columns** tab. Schema reconciliation is performed on this data.

If you set Generate SQL to No, you must specify your SQL statements at design-time. You must specify a value for **Table name** only if you also want to generate Data Definition Language (DDL) statements and you set **Table action** to Create, Drop, or Truncate.

The default value is No.

In some job configurations, the **Table name** property is disabled and you cannot provide a table name value. If you either generate a data definition language (DDL) SQL statement by clicking Generate, or generate the data manipulation language (DML) SQL statement by selecting **Tools > Generate SQL statement**, when the **Table name** property is disabled, then the generated SQL statement uses the placeholder <TABLENAME> instead of the table name. You can then replace <TABLENAME> with the actual table name.

Generate truncate statement at runtime
Use the properties in this group to determine whether you manually create or automatically generate a TRUNCATE TABLE statement at run time.

The Generate truncate statement at runtime property is available only if you set the following property values:

- **Write mode** to Insert
- **Table action** to Truncate

  The default value is No.

  Valid values are determined by the specific database.

**Insert statement**

Use this property to specify the SQL statement to insert rows into an existing table.

This property is available only if you set **Generate SQL** to No.

The following is an example of an INSERT statement where Name is the column in this statement and table1 is the name of the table. You must precede the column name of every target column name in the table with the word ORCHESTRATE that is followed by a period. ORCHESTRATE can be capitalized or typed in lowercase letters.

```
insert into table1(Name) values(ORCHESTRATE.Name)
```

This property is available only if you set **Write mode** to Insert, Insert then update, Update then insert, or Delete then insert.

Click ![...](image) to open a statement window in which to type your SQL statement. Click **Build** to open the SQL builder, where you can graphically build your SQL statement. These buttons are available only when you click inside the property.

Valid values are determined by the specific database.

**Isolation level**

Use this property to specify the degree of isolation of this transaction or unit of work as compared to other concurrently running transactions or units of work.

Transactions or units or work are committed in a target stage when the end of wave marker is detected. The default isolation level value is defined by the database, or it can be specified in the data source. The default value in the connector is **Read uncommitted**.

The following values are available:

- **Default**
  - The connector uses the value from the database.

- **Read uncommitted**
  - The row that is read during a unit of work or transaction can be changed by other transactions. Rows that are changed by other transactions can also be read even if the change is not yet committed by that transaction.

- **Read committed**
  - The row that is read during a unit of work or transaction can be changed by other transactions. However, a row that is changed by another transaction cannot be read until it is committed by that transaction.

- **Repeatable read**
  - The row that is read during a unit of work is not changed until the unit of
work is complete. Also, any row that is changed by another transaction cannot be read until the changed row is committed by that transaction that changed that row.

**Serializable**
The row that is read or changed cannot be read or changed until the row is committed by the transaction that is reading or changing that row.

**Key column**
Use this property to specify the name of the column to use as the sort key.

If the connector uses multiple input links and you choose **Ordered** in the **Record ordering** field, use the **Key column** field to specify the field to use as the sort key.

**Logging**
Use this property to log the values that are in each column when an SQL statement fails to insert, update, or delete a row. Each node that fails to insert, update, or delete rows prints the first row that failed on that node.

The **Logging** property is not available when **Write mode** property is set to **Bulk load**.

The following properties are included in the **Logging** category.

- Log column values on first row error
- Log key values only
- Column delimiter

**Log column values on first row error**
Use this property to specify whether the connector logs column values for the first row that failed on each node.

The **Log column values on first row error** property is not available when **Write mode** property is set to **Bulk load**.

The supported values for the **Log column values on first row error** property are:

- No - The default value is **No**.
- Yes - Logs column values for the first row that failed on each node.

**Log key values only**
Use this property to specify whether the connector logs the values of all columns or only key columns.

The **Log key values only** property is available when you specify **Yes** for the **Log column values on first row error** property. The **Log key values only** property is not available when **Write mode** property is set to **Bulk load**.

The supported values for the **Log key values only** property are:

- No - The default value is **No**.
- Yes - Logs key column values only.
Log multiple matches

Use this property to specify if you want the connector to log a message when the lookup statement returns multiple matching records for the input key record. The checking is performed for each input record separately.

The Log multiple matches property is available in the ODBC connector stage when the stage is running in a server job and in the lookup mode of operation. In this mode one or more reference links connect the ODBC connector stage with the Transformer stage. Even if the lookup statement in the connector returns multiple rows, only the first row is provided by the connector on the reference link.

When the ODBC connector stage is running in a parallel job and in lookup, it is connected with a reference link to the Lookup stage, and the Lookup stage provides support for handling multiple lookup matches.

The supported values for the property are:

- None - No message is logged for multiple matches and the job continues.
- Informational - An information message is logged and the job continues.
- Warning - A warning message is logged and the job continues.
- Fatal - An error message is logged and the job stops.

Null order

Use this property to specify where to place null values in the sort order.

If the connector uses multiple input links and you choose Ordered in the Record ordering field, use the Null order field to specify where to put null values in relation to non-null values. The choices are Before and After.

Partitioning method

Use this property to specify how to partition the data for parallel reads.

Each of these methods is an expression that determines how the records are divided into subsets for each processing node. The method that you use depends on how your data is distributed.

Both methods use the following information in their calculations:

- Total number of processing nodes
- Number of the current processing node
- Integer partitioning column value specified in the Column name property

This property is available only if you set Enable partitioning to Yes.

The following methods are available:

Modulus

The connector adds a modulus expression as a prefix to the WHERE clause of the SQL statement. This expression divides the value of the partitioning column by the total number of nodes or logical processors. When the remainder of this operation matches the number for the current processing node, the current node receives the record. The result is that each node receives a different subset of records.

This is the recommended choice when your data is not evenly distributed, such as: 1,2,3,4,5,6,7,8,9,10,20,40,80.
Min/max range
The connector adds a range expression as a prefix to the WHERE clause of the SQL statement. This expression checks the value of the partitioning column to determine whether it is within the unique range of values for each node or logical processor. The result is that each node receives a different subset of records. This is the default value.

This is the recommended choice when your data is evenly distributed, such as: 1,2,3,4,5,6,7,8,9,10,11,12,13.

Password
Use this property to specify the password for the user name that you specify in the Username property. These values are used for your data source connection.

This property might not be required, depending on the data source to which you connect.

Valid values are determined by the specific database up to a maximum of 256 characters.

Read After SQL statement from file
Use this property to specify the SQL statement that is run after any parallel processing occurs once per job.

This property has been added as a sub-property of the After SQL property. The Read After SQL statement from file property can be set to Yes or No. The default is No.

If you set this property to No, the behavior of the After SQL property is unaffected. You need to enter the SQL statements that need to be executed in the After SQL property.

If you set this property to Yes, you need to enter the absolute path to a file that contains the SQL statements that need to be executed in the After SQL property.

Read After SQL (node) statement from file
Use this property to specify the SQL statement that is run after all the data is processed once per node or logical processor, which can be a computer or a partition on a computer.

This property has been added as a sub-property of the After SQL (node) property. The Read After SQL (node) statement from file property can be set to Yes or No. The default is No. The property is only available when using the ODBC connector stage in a parallel job.

If you set this property to No, the behavior of the After SQL (node) property is unaffected. You need to enter the SQL statements that need to be executed in the After SQL (node) property.

If you set this property to Yes, you need to enter the absolute path to a file that contains the SQL statements that need to be executed in the After SQL (node) property.
Read Before SQL statement from file
Use this property to specify the SQL statement that is parsed and run before any processing occurs once per job.

This property has been added as a sub-property of the Before SQL property. The Read Before SQL statement from file property can be set to Yes or No. The default is No.

If you set this property to No, the behavior of the Before SQL property is unaffected. You need to enter the SQL statements that need to be executed in the Before SQL property.

If you set this property to Yes, you need to enter the absolute path to a file that contains the SQL statements that need to be executed in the Before SQL property.

Read Before SQL (node) statement from file
Use this property to specify the SQL statement that is run before any data is processed, once per node or logical processor.

This property has been added as a sub-property of the Before SQL (node) property. The Read Before SQL (node) statement from file property can be set to Yes or No. The default is No. The property is only available when using the ODBC connector stage in a parallel job.

If you set this property to No, the behavior of the Before SQL (node) property is unaffected. You need to enter the SQL statements that need to be executed in the Before SQL (node) property.

If you set this property to Yes, you need to enter the absolute path to a file that contains the SQL statements that need to be executed in the Before SQL (node) property.

Read select statement from file
Use this property to read the data from the file.

This property has been added as a sub-property of the Select statement property, which is only available when reading data using the ODBC connector as a source.

The Read select statement from file property can be set to Yes or No. The default is No.

If you set this property to No, the behavior of the Select statement property is unaffected. You have to enter the required SQL SELECT statement in the Select statement property that needs to be executed.

If you set this property to Yes, you have to enter the absolute path of the file in the Select statement property that contains the SQL SELECT statement which needs to be executed.

Record count
Use this property to specify the number of records to process before the connector commits the current transaction or unit of work.
You must specify a value that is a multiple of the value that you set for **Array size**. The default value is 20000. If you set **Record count** to 0, all available records are included in the transaction.

Valid values are integers between 0 and 999999999.

**Record ordering**

Use this property to specify how to process records across multiple links.

Specify how to process records from multiple links. Choose one of the following:

- **All records** - All records from the first link are processed; then all records from the second link are processed; and so on.
- **First record** - One record from each link is processed until all records from all links have been processed.
- **Ordered** - Records are selected from the input links based on the order that you specify by using the **Key column**, **Sort order**, and **Null order** fields.

**Row limit**

Use this property to specify the maximum number of rows that the ODBC connector will return.

In IBM InfoSphere Data Click, you can limit the number of rows that are offloaded from source to target. In the ODBC connector, this is accomplished with a new **Row limit** property used to specify the number of rows that the connector will return.

The default value is 200. If you set the **Row limit** to 0, all the rows that are produced by the database query are returned.

Valid values are integers in the range 0-999999999.

**Schema reconciliation**

Use the properties in this group to specify how the schema of the external resource is compared to the design-time schema. The external schema is either from the source in an output link or the target in an input link. The design-time schema is the schema that is represented on the Columns tab.

In this comparison, the schema metadata is compared, not the actual data.

**Select statement**

Use this property to specify the SQL statement to select rows from an existing table.

This property is available only if you set **Generate SQL** to No.

Click the browse button to open a statement window in which to type your SQL statement. Click **Tools**, then select **Build New SQL** to open the SQL builder, where you can graphically build, validate or generate your SQL statement. Both of these buttons are available only after you click inside this property value.

Valid values are determined by the specific database.
**Subtotal statement usage**

To use a subtotal in your SELECT statement, you must add a subtotal column to your schema on the **Columns** tab. For example, say that you want to use the following SELECT statement:

```
Select Title, sum(Age) AS subtotal from YOUR_TABLE GROUP BY Title
HAVING SUM(Age) >60 ORDER BY subtotal
```

You must define Title and subtotal as your column names on the **Columns** tab. When the connector executes this statement, Age is replaced by AS subtotal.

**LOB column usage**

Some ODBC drivers, such as Microsoft SQL Server drivers, have a limitation when you are working with large object (LOB) columns. With the SQL Server drivers, if a select list contains a LOB column, the LOB column must be listed last in the select list. For these drivers, there are two different scenarios where the query fails:

- The LOB columns are not listed last.
- You specify a wildcard asterisk (*) for the select list and the position of the LOB column in the table is not the last column in that table.

When you use drivers that have this limitation and you have LOBs in your table, you must ensure that the LOB columns are listed last in the select list.

In this example, you have two LOB columns in your Employees table: Description and Comments. If you write the following SELECT statement and run your job, the job fails:

```
select EmployeeID, Description, Comments, Photo from Employees
```

This is because the select list includes the Photo column after the LOB columns. Edit this statement to move the LOBs to the end as specified in the following statement:

```
select EmployeeID, Photo, Description, Comments from Employees
```

The LOB columns are now at the end of the statement. The job will run successfully.

**Session**

Use the properties in this group to configure various options for the connector.

These options apply to the current session, which is the time that begins when a connection to the database or data source is made and continues until the connection ends. This is the same as the length of time that is required to process a job.

**Sort order**

Use this property to specify whether to sort values in ascending or descending order.

If the connector uses multiple input links and you choose **Ordered** in the **Record ordering** field, use the **Sort order** field to specify whether to sort values in ascending or descending order.
SQL

Use the properties in this group to define the SQL statement or to define a combination of SQL statements.

Table action

Use this property to specify how to create new tables, to insert rows, or to edit rows in an existing destination table.

For the Append and Truncate values, the destination table must already exist. The default value is Append.

The following values are available:

Append
The schema of the data set must be compatible with the schema of the table.

New rows are appended to the existing destination table. To use this option, you must have INSERT privileges on the existing table.

Create
A new destination table is created. To use this option, you must have TABLE CREATE privileges on the existing table. You must provide the name of the destination table in the Table name property.

The schema of the data set determines the schema of the new table. The table is created with default properties. You can view this schema on the Columns tab of the stage editor. You can manually modify the generated SQL statement, or you can delete this generated statement and type your own.

If a table already exists with the same name as the table that you want to create, the operation terminates, and an error message is generated if you set Fail on error to Yes. No new table is created.

If you set Fail on error to No and a table with the same name already exists, the existing table is used. A warning message is generated, and the job continues to run.

Replace
The table that is specified as the destination table is dropped, and a new table is created in its place. To use this option, you must have INSERT and TABLE CREATE privileges on the existing table.

If a table does not already exist, but if you set Fail on error for Drop table to No, the job continues to run. If a table already exists with the same name as the table that you want to replace, the existing table is overwritten. The schema of the data set determines the schema of the new table.

Truncate
The attributes of the destination table are retained. However, existing records are discarded, and new records are appended. To use this option, you must have INSERT privileges on the existing table.

Table name for Partitioning method

Use this property to specify the table that contains the column that is queried to partition data when the Min/Max range partitioning method is used.

This property is available only if you set Partitioning method to Min/Max range.
Table name for Table action

Use this property to specify the name of the table for the SQL statement that the connector generates.

In the source context or lookup scenario, you must type a value for this property if you set Generate SQL to Yes. The connector uses this value for the table name in the SELECT statement that is generated at run time.

In the target context, you must type a value in this property under either of the following conditions:
- You set Generate SQL to Yes.
- You set Generate SQL to No and Table action to Create, Replace, or Truncate.

The connector uses this value for the table name in the INSERT, UPDATE, DELETE, CREATE TABLE, DROP TABLE, or TRUNCATE TABLE statement that is generated at run time.

In some job configurations, the Table name property is disabled and you cannot provide a table name value. If you either generate a data definition language (DDL) SQL statement by clicking Generate, or generate the data manipulation language (DML) SQL statement by selecting Tools > Generate SQL statement, when the Table name property is disabled, then the generated SQL statement uses the placeholder <TABLENAME> instead of the table name. You can then replace <TABLENAME> with the actual table name.

Transaction

Use the properties in this group to tune parameters for transactional processing, including end of wave markers, for this connector.

These options apply to the current transaction, which is a logical unit of work that is performed on a database management system. A transaction occurs within a session.

Truncate statement

Use this property to specify the SQL statement to truncate an existing database table.

This property is available only if you set the following property values:
- Write mode to Insert
- Table action to Truncate
- Generate truncate statement at runtime to No

Click Generate to automatically generate a generic SQL statement. You can manually modify the generated statement, or you can delete the generated statement and type your own statement.

Valid values are determined by the specific database.

Update statement

Use this property to specify the SQL statement to update an existing database table.

This property is available only if you set Generate SQL to No.
The following is an example of an UPDATE statement where Name is the column in this statement and table1 is the name of the table. You must precede the column name of every target column name in the table with the word ORCHESTRATE that is followed by a period. ORCHESTRATE can be capitalized or typed in lowercase letters.

update table1 set Name=ORCHESTRATE.Name where ID=ORCHESTRATE.ID

This property is available only if you set Write mode to Update, Insert then Update, or Update then Insert.

Valid values are determined by the specific database.

**Username**

Use this property to specify the user name for connections to the data source.

This property might not be required, depending on the data source to which you connect.

Valid values are determined by the specific database up to a maximum of 256 characters.

For more information about connection values for specific databases and database drivers, see *DataDirect Connect DataDirect Connect XE for ODBC User’s Guide and Reference* (odbcref.pdf).

**Write mode**

Use this property to specify the way that records from the data source are inserted into the destination table.

If records cannot be inserted or updated in the destination table and you have reject conditions on the reject link, the records are rejected and passed to the stage that is specified to receive the rejected records. The default value is Insert.

The following values are available:

**Insert** You can create an INSERT statement in the Insert statement property.

**Update** You can create a UPDATE statement in the Update statement property.

**Delete** You can create a DELETE statement in the Delete statement property.

**Insert then update** You can create an UPSERT statement by creating the INSERT statement in the Insert statement property and then by creating the UPDATE statement in the Update statement property. The INSERT statement is run before the UPDATE statement. The UPDATE statement is run on only those records that fail to be inserted.

**Update then insert** You can create an UPSERT statement by creating the UPDATE statement in the Update statement property and then by creating the INSERT statement in the Insert statement property. The UPDATE statement is run before the INSERT statement. The INSERT statement is run on only those records that fail to be updated.
Delete then insert
You can create an UPSERT statement by creating the INSERT statement in the Insert statement property and then by creating the DELETE statement in the Delete statement property. The DELETE statement is run before the INSERT statement.
Chapter 6. ODBC Enterprise stage

You can use the ODBC Enterprise stage to connect to all the major, external data sources from the parallel canvas of the InfoSphere DataStage: IBM DB2, IBM Informix®, Oracle, Sybase, SQL Server, Netezza, Teradata, MySQL, MS Access, and so on.

When you use IBM InfoSphere DataStage to access external data sources, you can choose from a collection of connectivity options. For most new jobs, use the ODBC Connector stage, which offers better functionality and performance than the ODBC Enterprise stage.

If you have jobs that use the ODBC Enterprise stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

ODBC enterprise stage operations

You can use an ODBC enterprise stage to perform database operations that are listed below:

- Read data from a data source
- Write data to a table in an external database
- Upsert data to an external database
- Lookup data in a data source

Read operation

Use the ODBC enterprise stage in read mode to read data from an external data source table and place them in an IBM InfoSphere DataStage output resource such as a data set or sequential file. The graphic below is a sample InfoSphere DataStage job that includes an ODBC enterprise stage in read mode.

Write operation

Use the ODBC enterprise stage in write mode to configure a connection to an external data source and insert records into a table. The stage takes records from a single input resource such as a data set, sequential file, and so on. The write mode determines how the records of a data set are inserted into the table. The graphic below is a sample IBM InfoSphere DataStage job that includes an ODBC enterprise stage in write mode.
Upsert operation

Use the ODBC enterprise stage in **upsert** mode to perform the upsert operations listed below:

- Insert records into an external data source table
- Update records in an external data source table
- Delete records from an external data source table
- Insert and then update records in an external data source table
- Update and then insert records into an external data source table

You can match records based on field names. The graphic below is a sample IBM InfoSphere DataStage job that includes an ODBC enterprise stage in upsert mode.

---

Lookup operation

Use the ODBC enterprise stage in **lookup** mode to perform a join operation between one or more external data source tables and an IBM InfoSphere DataStage input resource such as a data set or a sequential file. The data output is an InfoSphere DataStage data set. The graphic below is a sample InfoSphere DataStage job that includes an ODBC enterprise stage in lookup mode.
Working with ODBC enterprise stage

About this task

To use an ODBC enterprise stage in an IBM InfoSphere DataStage job, you must set appropriate stage properties. You can edit the stage properties by using options that the stage editor provides. To learn about the stage editor in detail, see Chapter 3, Stage Editors in the IBM InfoSphere DataStage and QualityStage Parallel Job Developer’s Guide.

To open the stage editor

- Double-click the data source or output resource icon on the InfoSphere DataStage parallel job designer canvas.

Stage Page

You can use this page to specify general information about your ODBC enterprise stage. Use the General, Advanced, and NLS Map tabs to specify appropriate information.

General tab

On the General tab, you can optionally specify a description of the ODBC enterprise stage.

Advanced tab

The Advanced tab properties determine how the ODBC enterprise stage behaves. Some advanced properties are set automatically and you cannot change the values of such properties. The properties are described below.

- **Execution mode.** This property determines whether your IBM InfoSphere DataStage job runs in parallel or sequential mode. The ODBC enterprise stage automatically sets the execution mode, and does not allow you to change this
value. If the stage is operating on only one file and there is one reader, the execution mode is sequential. Otherwise, the execution mode is parallel.

- **Combinability mode.** The default combinability mode is Auto. In Auto mode, the operators underlying the parallel stages are combined so that they run in the same process to improve performance.

- **Preserve partitioning.** You can select Set or Clear. If you select Set, the read operation sends a request to the next stage to preserve partitioning.

- **Node pool and resource constraints.** Select this option to constrain parallel execution to the node pools and resource pools that are specified in the grid. Use the grid to select node pools and resource pools. The selections are populated from the configuration file.

- **Node map constraint.** Select this option to constrain parallel execution to the nodes in a defined node map. You can define a node map by typing node numbers into the text box or by clicking the Browse button to select nodes from the Available Nodes window.

### NLS Map tab

On the NLS Map tab, you can define a character set map for the ODBC enterprise stage. The character set map that you define overrides the default character set map for the project or the job. If a job requires the NLS map as a parameter, you can specify on this tab that the map should be supplied as a job parameter.

### Input Page for write and upsert operations

You can use the Input page to specify exactly how you want data to be transferred from the ODBC enterprise stage to a remote host by using the ODBC protocol. You can use the General, Properties, Partitioning, Columns and Advanced tabs, and the Columns button to set appropriate properties.

The use of General, Partitioning, Columns, and Advanced tabs is similar for write and upsert operations. You must specify different values under the Properties tab and Columns button for write and upsert operations.

#### Input page for write operation

Use the tabs and buttons on this page to set up a write operation.

**General tab**

On the General tab, you can specify an optional description of the input link.

**Properties tab**

On the Properties tab, you can specify properties for the input link. The properties that you specify determine the data source, the operation to be performed on the data, the output resource, and so on. Properties without default settings appear in red and turn black when you supply a value. The following table lists the properties and their attributes. A more detailed description of each property follows.
The following table shows the properties and values on the Properties page for a write operation. Rows that contain a value in only the Category and property column are categories.

<table>
<thead>
<tr>
<th>Category and property</th>
<th>Value</th>
<th>Default</th>
<th>Required?</th>
<th>Dependent on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>String</td>
<td>N/A</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Write Method</td>
<td>Write</td>
<td>Write</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Write Mode</td>
<td>Append</td>
<td>Append</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Create</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Replace</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Truncate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data source</td>
<td>Data source</td>
<td>N/A</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>User</td>
<td>User name</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Password</td>
<td>Password</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert Array Size</td>
<td>Integer</td>
<td>2000</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Truncate Column Name</td>
<td>False</td>
<td>False</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>True</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open Command</td>
<td>SQL statement</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Close Command</td>
<td>SQL statement</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Length to Truncate</td>
<td>Integer</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Isolation Level</td>
<td>Read Uncommitted</td>
<td>Read Uncommitted</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Read Committed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Repeatable Read</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Serializable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Statement</td>
<td>SQL statement</td>
<td>N/A</td>
<td>No</td>
<td>Write method &gt; Write, Write mode &gt; Write method &gt; Upsert</td>
</tr>
<tr>
<td>Drop Unmatched Field</td>
<td>False</td>
<td>False</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>True</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Target**

Specify the **Table**, **Write Method** and **Write Mode** values here.
- **Table**: Specify the appropriate value here to connect the ODBC enterprise stage to a target file located in a remote host.
• **Write Method:** Set this property to **Write** to write and export data into a single table.

• **Write Mode:** Specify the appropriate value here to define how the records from the data source are inserted into the destination table. The **Write** mode can have one of the following values. Note that each of the below modes requires specific user privileges.
  - **Append** - This is the default mode. The **Append** mode requires that the destination table exists and the record schema of the data set is compatible with the schema of the table. In this mode, the **write** operation appends new rows to the existing destination table. The schema of the existing table determines the input interface of the stage.
  - **Create** - In this mode, the **write** operation creates a new destination table. If a table exists with the same name as the one being created, the operation terminates and an error message is displayed. The schema of the InfoSphere DataStage data set determines the schema of the new table. The table is created with simple default properties. To create a table with any properties other than the default properties, such as partitioned, indexed, or in a non-default table space, use the **-createstmt** option with your own **-createtable** statement.
  - **Replace** - In this mode, the **write** operation drops the table and creates a new one if the table with the same name exists. If a table with the specified name does not exist, the write operation creates a new table. The schema of the InfoSphere DataStage data set determines the schema of the new table.
  - **Truncate** - This mode requires an destination table. In this mode, the **write** operation retains the attributes of the destination table, but discards existing records and appends new records. The schema of the existing table determines the input interface of the ODBC enterprise stage.

**Connection**

Under this category, you specify values for the **Data source**, **Password** and **User** fields.

• **Data source:** This is a required field. Specify the database connection in this field by using any one of the methods below:
  - **Method 1:** Enter the data source name in the **Data source** field on the right side of the Properties page.
  - **Method 2:** Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then **Insert Job Parameters**.

    A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click **[New...]** from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

**Note:** Using InfoSphere DataStage Administrator, you can create parameters at the project level for all jobs within the project.

• **Password:** This is an optional field. Specify in this field the password for connecting to the data source by using any one of the methods below:
  - **Method 1:** Enter the data source name in the **Password** field on the right side of the Properties page.
  - **Method 2:** Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then **Insert Job Parameters**.
A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [New...] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

- **User**: This is an optional field. Specify in this field the user name for connecting to the data source using any one of the methods below:
  - **Method 1**: Enter the data source name in the **User** field on the right side of the Properties page.
  - **Method 2**: Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then **Insert Job Parameters**.

A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [New...] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

**Note**: If you have inserted all or some of the **Connection** category values from the job parameter list popup, then the job for which you provided these specifications takes the environment variables from the operating system. At runtime, you are prompted to modify the values for those environment variables.

**Options**

Under this category, you specify values for **Insert Array Size**, **Truncate Column Names**, **Close Command**, **Length to Truncate**, **Drop unmatched field**, **Open Command** and **Isolation level** properties. Under the **Options** category, the **Truncate Column Names** property appears by default. You can add the other properties mentioned above from the **Available properties to add** list.

- **Insert Array Size**: In this field, you specify the size of the insert host array. You can only enter an integer in this field. The default value is 2000.
- **Truncate Column Names**: You can set any of two values below, depending upon your requirement:
  - **True** - Set this value to indicate that column names will be truncated to the size allowed by the ODBC driver.
  - **False** - Set this value to disable truncating of column names.
- **Close Command**: Enter the SQL statement to be executed after an insert array is processed. This statement is executed only once on the conductor node.
- **Length to Truncate**: Specify the appropriate value for this option to indicate the length at which you want column names to be truncated.
- **Drop unmatched field**: You can set one of the values below, depending upon your requirement:
  - **True** - Set this value to indicate that unmatched columns in the data set should be dropped. An unmatched column is a column for which there is no identically named column in the data source table.
  - **False** - This is the default value. This value indicates that unmatched fields of the data set will not be dropped.
- **Open Command**: Enter the SQL statement to be executed before the insert array is processed. This statement is executed only once on the conductor node.
- **Isolation level**: Select the isolation level for accessing data from five available options:
  - **Read**
The database specified in the data source determines the default isolation level.

**Partitioning tab**

On the **Partitioning** tab, you can specify details about how the incoming data is partitioned or collected before it is written to the destination database. You can also specify that the data should be sorted before it is written to the destination database.

By default, the ODBC enterprise stage partitions data in Auto mode. Auto mode provides the best partitioning method depending on the execution mode settings of the current and preceding stages and how many nodes are specified in the configuration file.

If the stage is operating in sequential mode, it collects data before writing it to the destination database by using the default **Auto** collection method. You can use the **Partitioning** tab to override this default behavior. The options that you set on this tab behave differently according to whether the current and preceding stages are set to run in parallel or sequential mode.

**Columns tab**

On the **Columns** tab, you can view and modify column metadata for the input link. Use the **Save** button to save any modifications that you make in the column metadata. Use the **Load** button to load an existing source table.

From the Table Definitions window, select the appropriate table to load and click **OK**. The Select Column dialog is displayed. To ensure appropriate conversion of data types, clear the **Ensure all Char columns use Unicode** check box.

**Advanced tab**

On the **Advanced** tab, you can specify how input data for the ODBC enterprise stage is buffered. By default, data is buffered so that no deadlocks can arise. A deadlock is a situation in which a number of stages are mutually dependent. An example is a situation in which one or more stages are waiting for input from another stage, and those stages cannot output data until they have received input.

**Columns button**

Use the **Columns** button to define column names for the destination table.

**Input page for upsert operation**

For an upsert operation, you need to specify appropriate values on the Properties tab and Columns button.

On the **Properties** tab, you can specify properties for the input link. The properties that you specify determine the data source, the operation to be performed on the
data, the output resource, and so on. Properties without default settings appear in red and turn black when you supply a value. The following table lists the properties and their attributes.

Table 9. Input page properties and corresponding values for Upsert.

The following table shows the properties and values on the Properties page for an upsert operation. Rows that contain a value in only the Category and property column are categories.

<table>
<thead>
<tr>
<th>Category and property</th>
<th>Values</th>
<th>Default</th>
<th>Required?</th>
<th>Dependent on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>String</td>
<td>N/A</td>
<td>Yes</td>
<td>Upsert Method: Auto-generated SQL · SQL builder generated SQL</td>
</tr>
<tr>
<td>Write Method</td>
<td>Upsert</td>
<td>Write</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Upsert Mode</td>
<td>Delete only</td>
<td>Insert then Update</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Delete then Insert</td>
<td>Insert then Update</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Insert only</td>
<td>Insert then Update</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Update only</td>
<td>Update then Insert</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Update then Insert</td>
<td>Auto-generated SQL</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Insert SQL</td>
<td>SQL statement</td>
<td>N/A</td>
<td>No</td>
<td>Upsert Mode: Delete then Insert · Insert only · Insert then Update · Update then Insert</td>
</tr>
<tr>
<td>Delete SQL</td>
<td>SQL statement</td>
<td>N/A</td>
<td>No</td>
<td>Upsert Mode: Delete only · Delete then Insert</td>
</tr>
<tr>
<td>Update SQL</td>
<td>SQL statement</td>
<td>N/A</td>
<td>No</td>
<td>Upsert Mode: Update only · Insert then Update · Update then Insert</td>
</tr>
<tr>
<td>Connection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 9. Input page properties and corresponding values for Upsert (continued).

The following table shows the properties and values on the Properties page for an upsert operation. Rows that contain a value in only the Category and property column are categories.

<table>
<thead>
<tr>
<th>Category and property</th>
<th>Values</th>
<th>Default</th>
<th>Required?</th>
<th>Dependent on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data source</td>
<td>String</td>
<td>N/A</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>User</td>
<td>String</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Password</td>
<td>String</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert Array Size</td>
<td>Integer</td>
<td></td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Truncate Column Names</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• False</td>
<td>False</td>
<td></td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>• True</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Open Command</td>
<td>SQL statement</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Close Command</td>
<td>SQL statement</td>
<td>N/A</td>
<td>No</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Properties tab

Target

Under the Target category, specify values at least for the Auto-generate Upsert SQL, Upsert mode, and Write Method properties. Based on the values that you select for the Upsert mode and Auto-generate Upsert SQL properties, you see additional properties. Specify values for these additional properties depending on your requirement.

- **Auto-generate Upsert SQL**: Specify whether the SQL is automatically generated from the set of loaded columns. The default value for this property is No. When you set the value to Yes, an additional property Table appears. Specify the name of the destination table. You can specify this value by using the job parameter popup list.

- **Upsert mode**: Specify how the insert and update statements are to be derived. The way in which you set up the Upsert mode property depends upon the option you select for the Auto-generate Upsert SQL property. The table below describes how the Upsert mode options depending upon the Auto-generate Upsert SQL property you have selected. If you select Yes, the queries are generated from the built-in templates. If you select No, you can define queries yourself.
<table>
<thead>
<tr>
<th>Option</th>
<th>If Auto-generated Upsert SQL = Yes</th>
<th>If Auto-generated Upsert SQL = No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete only</td>
<td>Select this option to have a <em>delete</em> statement generated automatically, based on the table and column details that you provide. When you select <strong>Delete Only</strong> as the <strong>Upsert mode</strong>, an additional property <strong>Delete SQL</strong> appears on the list. You must select an appropriate value for the <strong>Delete SQL</strong> property. To view the automatically generated statement, click <strong>Delete SQL</strong>. The statement appears in a field on the right side of the list of properties.</td>
<td>Select this option to create your own <em>delete</em> statement. Click <strong>Delete SQL</strong>, and then enter the statement in the <strong>Delete SQL</strong> field on the right side of the list of properties.</td>
</tr>
<tr>
<td>Delete then Insert</td>
<td>Select this option to have <em>delete and insert</em> statements generated automatically, based on the table and column details that you provide. When you select <strong>Delete then Insert</strong> as the <strong>Upsert mode</strong>, two additional properties <strong>Delete SQL</strong> and <strong>Insert SQL</strong> appear on the list. You must select appropriate values for these two properties. To view the automatically generated statements, click <strong>Delete SQL</strong> and <strong>Insert SQL</strong>. The statements appear in their respective fields on the right side of the list of properties.</td>
<td>Select this option to create your own <em>delete and insert</em> statements. Click <strong>Delete SQL</strong> and then enter the statement in the <strong>Delete SQL</strong> field on the right side of the list of properties. Next, click <strong>Insert SQL</strong> and then enter the statement in the <strong>Insert SQL</strong> field on the right side of the list of properties.</td>
</tr>
<tr>
<td>Insert only</td>
<td>Select this option to have an <em>insert</em> statement generated automatically, based on the table and column details that you provide. When you select <strong>Insert only</strong> as the <strong>Upsert mode</strong>, an additional property <strong>Insert SQL</strong> appears on the list. You must select an appropriate value for this property. The statement appears in a field on the right side of the list of properties.</td>
<td>Select this option to create your own <em>insert</em> statement. Click <strong>Insert SQL</strong>, and then enter the statement in the <strong>Insert SQL</strong> field on the right side of the list of properties.</td>
</tr>
<tr>
<td>Option</td>
<td>If Auto-generated Upsert SQL = Yes</td>
<td>If Auto-generated Upsert SQL = No</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Insert then Update</td>
<td>Select this option to have insert and update statements generated automatically, based on the table and column details that you provide. When you select Insert then Update as the Upsert mode, two additional properties Insert SQL and Update SQL appear on the list. You must select appropriate values for these two properties. The statements appear in their respective fields on the right side of the list of properties.</td>
<td>Select this option to create your own insert and update statements. Click Insert SQL and then enter the statement in the Insert SQL field on the right side of the list of properties. Next, click Update SQL and then enter the statement in the Update SQL field on the right side of the list of properties.</td>
</tr>
<tr>
<td>Update only</td>
<td>Select this option to have an update statement generated automatically, based on the table and column details that you provide. The Update SQL property appears on the list by default. You must select an appropriate value for this property. The statement appears in a field on the right side of the list of properties.</td>
<td>Select this option to create your own update statement. Click Insert SQL, and then enter the statement in the Insert SQL field on the right side of the list of properties.</td>
</tr>
<tr>
<td>Update then Insert</td>
<td>Select this option to have update and insert statements generated automatically, based on the table and column details that you provide. When you select Update then Insert as the Upsert mode, two additional properties Update SQL and Insert SQL appear on the list. You must select appropriate values for these two properties. The statements appear in their respective fields on the right side of the list of properties.</td>
<td>Select this option to create your own update and insert statements. Click Update SQL and then enter the statement in the Update SQL field on the right side of the list of properties. Next, click Insert SQL and then enter the statement in the Insert SQL field on the right side of the list of properties.</td>
</tr>
</tbody>
</table>

- **Write Method:** To select the Upsert mode, set this value to Upsert.

**Connection**

Under this category, you specify values for Data source, Password and User.

- **Data source:** This is a required field. Specify the database connection in this field by using any one of the methods below:
  - **Method 1:** Enter the data source name in the Data source field on the right side of the Properties page.
- **Method 2**: Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then Insert Job Parameters.

  A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [New...] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

- **Password**: This is an optional field. Specify in this field the password for connecting to the data source by using any one of the methods below:
  - **Method 1**: Enter the data source name in the Password field on the right side of the Properties page.
  - **Method 2**: Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then Insert Job Parameters.

  A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [New...] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

- **User**: This is an optional field. Specify in this field the user name for connecting to the data source by using any one of the methods below:
  - **Method 1**: Enter the data source name in the User field on the right side of the Properties page.
  - **Method 2**: Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then Insert Job Parameters.

  A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [New...] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

**Note**: If you have inserted all or some of the Connection category values from the job parameter list popup, then the job for which you provided these specifications takes the environment variables from the operating system. At runtime, you are prompted to modify the values for those environment variables.

**Options**

Under the Options category, you specify values for Open Command, Close Command, Output Reject Records, and Insert Array Size properties. Under the Options category, the Output Reject Records property appears by default. You can add the other properties mentioned above from the Available properties to add list.

- **Open Command**: Specify in single quotes a command to be parsed and executed by the ODBC database on all processing nodes before the ODBC table is opened. You can specify this value as a job parameter.

- **Close Command**: Specify in single quotes a command to be parsed and executed by the ODBC database on all processing nodes after the ODBC enterprise stage completes processing the ODBC table. You can specify this value as a job parameter.

- **Output Reject Records**: Select one of the below values:
  - **True**: Select this value to indicate that the rejected records should be sent to the reject link.
- **False**: This is the default value. Select this value to indicate that rejected records should not be sent to the reject link.

- **Insert Array Size**: Specify the size of the insert host array. This property only accepts an integer. The default value is 2000.

**Note**: Using InfoSphere DataStage Administrator, you can also create parameters at the project level for all jobs within the project.

**Columns button**

Use the **Columns** button to define a list of column names for the destination table. If the ODBC enterprise stage editor for the Input page in **Read** or **Upsert** mode shows both input and output links, then you can choose one of them to function as the reject link.

A reject link contains raw data for columns rejected due to schema mismatch, after the SELECT statement that you specified is executed.

**Output page for read and lookup operations**

The Output page appears both in the **Read** and **Lookup** modes of the ODBC enterprise stage. Use the Output page to provide details about the output link to the ODBC enterprise stage from a remote host. Using the output link, you can access data from a remote host through ODBC. In the **Read** mode, the ODBC enterprise stage only has an output link. In the **Lookup** mode, the stage has an output link as well as a reference link that connects the stage to a lookup stage.

You can use the General, Properties, Partitioning, Columns and Advanced tabs, and the Columns button to set appropriate properties.

The use of General, Partitioning, Columns, and Advanced tabs is similar for read and lookup operations. You must specify different values under the Properties tab and Columns button for write and upsert operations.

**Output page for a read operation**

Use the tabs and buttons on this page to set up a read operation.

**General tab**

On the **General** tab, optionally enter a description for read operation.

**Properties tab**

On the Properties tab, specify the properties as follows.

**Source**

Under this category, you specify values for **Read Method**, **Query**, and **Table**.

- **Read Method**: Use this property to specify a table or a query for reading the ODBC database. The default value for **Read Method** is **Table**. If you choose **Table** as the read method, then you must specify the data source table for the **Table** option (see the **Table** section below for details). Alternatively, you can setup **Read Method** as an SQL query. In that case, you must specify whether you want the query to be generated automatically or you want to define the query yourself.
Note: The **Query** property appears on the list of properties only when you select **Auto-generated SQL** or **User-defined SQL** as the **read** method. To select one of these two types of query, click **Read Method**, and then select the appropriate option from the **Read Method** list on the right side of the properties list.

- **Query**: Specify an SQL statement for reading a table. The statement should specify the table to read and the processes to perform on the table during the **read** operation. This statement can contain **join** and **view** operations, database links, synonyms, and so on. Choose from the following available values:
  - **Auto-generated SQL**: Select this read method if you wish to have an SQL query generated automatically, based on the table that you specify in the **Table** field and the columns that you define.
  - **User-defined SQL**: Select this read method if you wish to define your own SQL query.

  *Note*: An SQL query is read both in parallel and sequential execution modes.

- **Table**: If you have chosen **Table** as the read method (see the **Read Method** section in the above), then you must specify the name of the source ODBC table. Note that the specified table must exist and you must have **SELECT** privileges for this table. If your ODBC user name does not correspond to the owner of the specified table, you can prefix it with a table owner. You must add a new job parameter to fix the table owner name.

**To fix the table owner name:**
1. Click **Table** and then the arrow on the right side of the dialog.
2. Click **Insert job parameter** and then [**New...**] from the popup list.
3. In the Job Properties dialog that appears, enter the required table details in the **Default Value** column for the **$user** parameter. Use the below format:
   
   ```
   table_owner.table_name
   ```
   
   Before you select a value for this option, you must fulfil the below dependencies:
4. Use the **WHERE** clause in your **SELECT** statement to specify the rows of the table to be included or excluded from the read operation. If you do not supply a **WHERE** clause, all rows are read.
5. You can specify in your **SELECT** statement the columns that you wish to read. You must specify the columns in this list in the same order as they are defined in the record schema of the input table.

**Connection**

Under this category, you specify the **Data source**, **Password** and **User** values.

- **Data source**: This is a required field. Specify the database connection in this field by using any one of the methods below:
  - **Method 1**: Enter the data source name in the **Data source** field on the right side of the Properties page.
  - **Method 2**: Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then **Insert Job Parameters**.

  A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [**New...**] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.
Password: This is an optional field. Specify in this field the password for connecting to the data source by using any one of the methods below:

- **Method 1**: Enter the data source name in the **Password** field on the right side of the Properties page.
- **Method 2**: Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then **Insert Job Parameters**.

A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [New...] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

User: This is an optional field. Specify in this field the user name for connecting to the data source by using any one of the methods below:

- **Method 1**: Enter the data source name in the **User** field on the right side of the Properties page.
- **Method 2**: Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then **Insert Job Parameters**.

A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [New...] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

**Note**: If you have inserted all or some of the **Connection** category values from the job parameter list popup, then the job for which you provided these specifications takes the environment variables from the operating system. At runtime, you are prompted to modify the values for those environment variables.

**Options**

Under this category, you specify values for **FetchArraySize**, **Isolation Level**, **Close Command** and **Open Commands**. All of these properties are optional. You see these properties in the **Available properties to add** list that appears in the bottom right corner of the Output page. To add any of these subproperties under **Options**, click **Options**, and then the property that you wish to add from the **Available properties to add** list.

- **Fetch Array Size**: Specify the number of rows to retrieve during each fetch operation. The default value is 1.
- **Isolation Level**: Enter the isolation level for accessing data. Choose from the four available options:
  - Read Committed
  - Read Uncommitted
  - Repeatable Read
  - Serializable

  The database that you specified for the **Data source** option (see the **Data source** section on the above) determines the default isolation level.

- **Close Command**: Enter an SQL statement to be executed after the insert array is processed. You cannot commit work by using this option. The statements are executed only once on the conductor node.
- **Open Command**: Enter an SQL statement to be executed before the insert array is processed. The statements are executed only once on the conductor node.
• **Partition Column**: Specify the partition column name to read data in parallel mode. The prerequisite for a parallel read is that the data must be in increasing order. Reading data in parallel mode enhances the performance of the read operator.

  **Note**: Using InfoSphere DataStage Administrator, you can create parameters at the project level for all jobs within the project.

**Advanced tab**

Use the **Advanced** tab to specify how input and output data for the ODBC enterprise stage is buffered. By default, data is buffered so that no deadlocks can arise. A deadlock is a situation in which a number of stages are mutually dependent, and cannot output data until they have received input.

**Columns button**

Use the **Columns** button to define a list of column names for the output table.

**View Data button**

To view the number of rows in the table that you specified for the **Table** option under **Source**, click the **View Data** button. You can specify the number of rows that you wish to view at a time.

**Output page for a lookup operation**

For a lookup operation, you need to specify appropriate values on the **Properties** tab. You can use the rest of the tabs in the same manner as for the read operation.

**Properties tab**

Use the **Properties** tab to set appropriate values for **Source**, **Connection**, and **Options** properties. Below is a sample of the dialog that appears when you click the **Properties** tab from the Output page.

**Source**

Under this category, you specify values for **Lookup Type**, **Query**, **Read Method**, and **Table**.

• **Lookup Type**: You can choose **Normal** or **Sparse** as the lookup method.
  
  – **Normal**: This is the default lookup method. A normal lookup is an in-memory lookup on an ODBC database table. In case of a normal lookup, the lookup stage can have multiple reference links.
  
  – **Sparse**: A sparse lookup accesses the source database directly. In case of a sparse lookup, the lookup stage has one reference link.

• **Query**: Specify an SQL statement for reading a table. The statement should specify the table to read and the processes to perform on the table during the read operation. This statement can contain join and view operations, database links, synonyms, and so on. Choose from the following available values:

  – **Auto-generated SQL**: Select this read method if you wish to have an SQL query generated automatically, based on the table that you specify in the **Table** field and the columns that you define.
  
  – **User-defined SQL**: Select this read method if you wish to define your own SQL query.
Note: An SQL query is read in both parallel and sequential execution modes.

- **Table:** If you have chosen **Table** as the read method (see the Read Method section above), then you must specify the name of the source ODBC table. Note that the specified table must exist and you must have **SELECT** privileges for this table. If your ODBC user name does not correspond to the owner of the specified table, you can prefix it with a table owner. You must add a new job parameter to fix the table owner name.

**To fix the table owner name:**
1. Click **Table** and then the arrow on the right side of the dialog.
2. Click **Insert job parameter** and then [New...] from the popup list.
3. In the Job Properties dialog that appears, enter the required table details in Default Value column for the $user parameter. Use the below format:

   ```
   table_owner.table_name
   ```

   Before you select a value for this option, you must fulfill the below dependencies:

4. Use the WHERE clause in your **SELECT** statement to specify the rows of the table to be included or excluded from the read operation. If you do not supply a WHERE clause, all rows are read.
5. You can specify in your **SELECT** statement the columns that you wish to read. You must specify the columns in this list in the same order as they are defined in the record schema of the input table.

**Connection**

Under this category, you specify the **Data source**, **Password** and **User** values.

- **Data source:** This is a required field. Specify the database connection in this field by using any one of the methods below:
  - **Method 1:** Enter the data source name in the **Data source** field on the right side of the Properties page.
  - **Method 2:** Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then **Insert Job Parameters**.

A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [New...] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

Using the IBM InfoSphere DataStage and QualityStage Administrator, you can also create parameters at the project level for all jobs within the project.

- **Password:** This is a required field. Specify in this field the password for connecting to the data source by using any one of the methods below:
  - **Method 1:** Enter the data source name in the **Password** field on the right side of the Properties page.
  - **Method 2:** Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then **Insert Job Parameters**.

A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [New...] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

Using the InfoSphere DataStage and QualityStage, you can also create parameters at the project level for all jobs within the project.
• **User:** his is a required field. Specify in this field the user name for connecting to the data source by using any one of the methods below:
  
  – **Method 1:** Enter the data source name in the **User** field on the right side of the Properties page.
  
  – **Method 2:** Insert the desired value as a job parameter. Click the pointer button on the extreme right side of the Properties page, and then **Insert Job Parameters**.
    
    A popup appears with a list of available job parameters from which you can choose. If you wish to create a new parameter for the job, click [New...] from the popup list, and create an appropriate environment variable by using the Job Properties dialog that appears.

    Using the InfoSphere DataStage and QualityStage, you can also create parameters at the project level for all jobs within the project.

    **Note:** If you have inserted all or some of the **Connection** category values from the job parameter list popup, then the job for which you provided these specifications takes the environment variables from the operating system. At runtime, you are prompted to modify the values for those environment variables.

**Options**

Under this category, specify the **Fetch Array Size**, **Isolation Level**, **Close Command** and **Open Commands**. All of these properties are optional. You see these properties in the **Available properties to add** list that appears in the bottom right corner of the Output page. To add any of these subproperties under **Options**, click **Options**, and then the property that you wish to add from the **Available properties to add** list. Note that Isolation Level appears on the list only if you select **Lookup Type** as **Normal**.

• **Fetch Array Size:** Specify the number of rows to retrieve during each **fetch** operation. The default value is 1.

• **Isolation Level:** Enter the isolation level for accessing data. Choose from the four available options:
  
  – Read Committed
  – Read Uncommitted
  – Repeatable Read
  – Serializable

  The database that you specified for the **Data source** option (see the **Data source** section above) determines the default isolation level.

• **Close Command:** Enter an SQL statement to be executed after the insert array is processed. You cannot commit work by using this option. The statements are executed only once on the conductor node.

• **Open Command:** Enter an SQL statement to be executed before the insert array is processed. The statements are executed only once on the conductor node.

**Output page with a reject link for an upsert operation**

If the stage editor shows both input and output links for a lookup or write operation, you can choose any of the links as the reject link. A reject link contains raw data for columns rejected due to schema mismatch, after your **SELECT** statement is executed.
An Output page with a reject link or an upsert operation contains the same tabs and buttons as the Output page for a read or lookup operation. You can use these tabs and buttons in the same manner as you did for a read or lookup operation.
Chapter 7. ODBC stage

The ODBC stage is used to represent a database that supports the industry-standard Open Database Connectivity API. The ODBC stage is a passive connectivity stage.

When you use IBM InfoSphere DataStage to access external data sources, you can choose from a collection of connectivity options. For most new jobs, use the ODBC Connector stage, which offers better functionality and performance than the ODBC stage.

If you have jobs that use the ODBC stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

Using ODBC Stages

You can use an ODBC stage to extract, write, or aggregate data. Each ODBC stage can have any number of inputs or outputs. Input links specify the data that you are writing. Output links specify the data that you are extracting and any aggregations that are required.

You can specify the data on an input link by using any of the following methods:

- an SQL statement that is constructed by IBM InfoSphere DataStage
- a user-defined SQL query
- a stored procedure

You can specify the data on an output link by using any of the following methods:

- an SQL statement that is processed by ODBC
- an SQL statement that is passed through ODBC and processed by the underlying database
- a stored procedure

When you edit an ODBC stage, the ODBC stage editor opens. This window can have up to three pages (depending on whether there are inputs to and outputs from the stage):

Stage  Displays the name of the stage you are editing. Use the General tab to define the data source name and to type a description of the stage in the Description field. You can also specify a quote character and schema delimiter that are used by the data source, or you can retrieve them automatically by clicking the Get SQLInfo button. Use the NLS tab to define a character set map to use with the stage, if required. For details, see “Defining Character Set Maps” on page 87.

Inputs  This page is displayed only if you have an input link to the stage. Specifies the SQL tables or stored procedure 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 that is used to write the data.

Outputs  This page is displayed only if you have an output link from the stage. Specifies the SQL tables or stored procedure to use, and the associated
column definitions for each data output link. This page also contains the SQL SELECT statement or call syntax that is used to extract the data.

Click OK to close the ODBC stage editor. Changes are saved when you save the job design.

**Must Do's**

**About this task**

This section specifies the minimum steps needed to get an ODBC stage functioning.

To edit an ODBC stage:

**Procedure**

1. Define the connection.
2. If NLS is enabled, optionally define a character set map on the NLS tab.
3. Define the data on the input links.
4. Define the data on the output links.

**Results**

You perform these steps in the ODBC stage editor.

---

**Defining the Connection**

To connect to an ODBC data source, you must install and configure a suitable ODBC driver on your system.

**ODBC Connection Parameters**

**About this task**

The ODBC connection parameters are set on the General tab on the Stage page.

To connect to an ODBC data source:

**Procedure**

1. In the **Data source name** field, select the data source name. This list contains all of the data sources that are defined in the **Table Definitions > ODBC** folder in the repository.
   
   If the data source name that you want is not listed, type the name in the **Data source name** field or create a new table definition. You can also specify a job parameter.

2. In the **User name** field, type the name to use to connect to the data source. You can also specify a job parameter.

3. In the **Password** field, type the password to use. You can also specify a job parameter.

   **Note:** Certain ODBC drivers allow you to specify the user name and password to use on connection. If you are connecting to an ODBC data source by using a driver that has a user name and password already specified, you do not need to specify a user name and password on the General tab.
4. Optional: Specify the quote character that is used by the data source. By default this is set to " (double quotes). You can also click Get SQLInfo to connect to the data source and retrieve the quote character it uses. An entry of 000 (three zeroes) specifies that no quote character should be used.

5. Optional: Specify the schema delimiter that is used by the data source. By default this is set to . (period), but you can specify a different schema delimiter, or multiple schema delimiters. For example, where table identifiers have the form:
   Node:Schema.Owner;TableName
   you would type :. into this field. You can also click Get SQLInfo to connect to the data source and retrieve the schema delimiter it uses.

6. Optional: In the Description field, type a description of the ODBC stage.

---

### Defining Character Set Maps

Use the NLS tab on the Stage page to define a character set map for an ODBC stage. To change the default character set map that is defined for the project or the job, select a map name from the list. The NLS tab also has the following options:

- **Show all maps.** Lists all of the maps that are supplied with IBM InfoSphere DataStage. Maps cannot be used unless they have been loaded by using the Administrator.
- **Loaded maps only.** Displays the maps that are loaded and ready for use.
- **Allow per-column mapping.** Allows character set maps to be specified for individual columns within the table definition. If per-column mapping is selected, an extra property called NLS Map appears in the grid on the Columns tab.
- **Use Job Parameter.** Allows you to specify a character set map as a parameter to the job that contains the stage. If the parameter has not yet been defined, you are prompted to define it from the Job Properties window.

---

### Handling SQL Server Data Types

The ODBC stage can handle the following SQL Server data types:

- GUID
- Timestamp
- SmallDateTime

**GUID Type**

**About this task**

If you imported column metadata from an SQL Server data source that contains GUID types, then no special action is needed. Just load the required table definition on the Columns tab, and IBM InfoSphere DataStage will set up the GUID data types automatically.

If you are manually specifying the table definition, take the following steps for the column that you want to contain a GUID data type:

**Procedure**

1. Set the SQL Type to VarChar.
2. Set the length to 36.
3. Set the data element to SQL.GUID.
Results

Data will be read in as VARCHAR data type, and written to a database as UNIQUEIDENTIFIER data type.

Note: To read new UIDs from a table rather than existing ones, specify newid() in the Derivation field for that column on the Outputs page Columns tab.

Timestamp Type

About this task

Timestamp types cannot be handled automatically by importing the SQL Server metadata. They always need to be set up manually.

For the column that you want to contain a Timestamp data type:

Procedure

1. Set the SQL Type to Integer.
2. Set the length to 10.
3. Set the data element to SQL.ROWVERNUM.

Results

Data will be read in as an integer.

For pseudo-timestamp data:

About this task

(where the data is timestamp but the column in the database is defined as binary):

Procedure

1. Set the SQL Type to Integer.
2. Set the length to 10.
3. Set the data element to SQL.BINARY8.

Results

Data will be read in as an integer, and written to a database as a binary.

SmallDateTime Type

About this task

If you imported column metadata from an SQL Server data source that contains SmallDateTime types, no special action is needed. Just load the required table definition on the Columns tab, and the IBM InfoSphere DataStage will set up the SmallDateTime data types automatically.

If you are manually specifying the table definition, take the following steps for the column that you want to contain a SmallDateTime data type:

Procedure

1. Set the SQL Type to Char.
2. Set the length to 16.
3. Set the data element to SQL.SMALLDATETIME.
Results

Data will be read in as a 16-character string of the form yyyy-mm-dd hh:mm and written to a database as a SmallDateTime.

Defining ODBC Input Data

When you write data to a table (or a stored procedure) in an ODBC database, the ODBC stage has an input link. The properties of this link and the column definitions of the data are defined on the Inputs page in the ODBC stage editor.

The Inputs page has the following field and up to six tabs. The tabs that are displayed depend on the update action that you select on the General tab, and whether you want to create a table in the target database:

Input name

This field displays the name of the input link. Choose the link that you want to edit from the list. The list contains all of the input links to the ODBC stage.

General

This tab is displayed by default and contains the following components:

• Table name. This field appears when the update action is not Call stored procedure or User-defined SQL. It is the name of the table that the data is written to. Choose the table from the Table name list. This list contains all of the tables that are defined in the Table Definitions > ODBC > Data source folder in the repository. Data source is the data source name chosen on the General tab on the Stage page.

If the table that you want is not listed, you need to create a table definition. Alternatively, click Browse... to display the Table Definitions window and choose a suitable table definition. You can also specify a job parameter in this field.

• Stored procedure name. This field appears only when the update action is Call stored procedure. It is the name of the procedure that the data is passed to. Choose the stored procedure that you want to use from the list. This list contains all of the stored procedures that are defined in the Table Definitions > StoredProcedures folder in the repository for the specified DSN.

If the stored procedure that you want is not listed, you must define the stored procedure. Alternatively, click Browse... to search for the stored procedure.

• Update action. Specifies how the data is written. Choose the option you want from the list:

  – Clear the table, then insert rows. Deletes the contents of the table and adds the new rows.
  – Insert rows without clearing. Inserts the new rows in the table.
  – Insert new or update existing rows. New rows are added or, if the insert fails, the existing rows are updated.
  – 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. If a row with the supplied key does not exist in the table, then the table is not updated and a warning is logged.
- Update existing or insert new rows. The existing data rows are updated or, if this fails, new rows are added.

- Call stored procedure. Writes the data using a stored procedure. When you select this option, the Stored procedure name field appears.

- User-defined SQL. Writes the data by using a user-defined SQL statement. When you select this option, the View SQL tab is replaced by the Enter SQL tab.

- Create table in target database. Select this check box if you want to automatically create a table in the target database at run time. A table is created based on the defined column set for this stage. If you select this option, an additional tab, Edit DDL, appears. This shows the SQL CREATE statement to be used for table generation.

- Description. Contains an optional description of the input link.

Columns
This tab is always present and contains the column definitions for the data written to the table or file. If you are using a stored procedure, the column definitions represent the stored procedure input parameters. You must have at least the same number of column definitions as expected parameters. The column definitions are used in the order that they appear in the Columns grid.

View SQL
This tab appears when you select any update action other than User-defined SQL. It displays the SQL statement or stored procedure call syntax that is used to write the data. You cannot edit this statement, but you can click Copy to copy it to the Clipboard for use elsewhere.

Enter SQL
This tab appears only when you set the update action to User-defined SQL, when it replaces the View SQL tab. It displays the user-defined SQL statement.

Edit DDL
This tab appears if you have chosen to automatically generate a table at run time by selecting the Create table in target database check box on the General tab. It displays the SQL CREATE statement that will be used to create the table. To generate the statement, click Create DDL. InfoSphere DataStage will connect to the target database and generate the statement. (If you are creating a table in a Sybase database, the Sybase database needs to have the "Data definition language in transaction" option set.) You can edit the statement on this tab to make any required changes. This tab also allows you to specify that any existing table by this name should be dropped first. If you do not select this option and such a table already exists in the target database, then the create will fail.

Error Codes
This tab appears when you set the update action on the General tab to Call stored procedure (or if you have not yet set an update action). It allows you to specify a space-separated list of values in the Fatal errors and Warnings fields to handle raiserror calls within the stored procedure. You can also load predefined information from the repository by clicking Load.

Transaction Handling
This tab allows you to specify the transaction handling features of the stage as it writes to the ODBC data source. You can choose whether to use transaction grouping or not, and you can specify an isolation level, the
number of rows written before each commit, and the number of rows written in each operation. A grid shows details of the transaction group to which the currently selected input link belongs. For more information, see "Specifying Transaction Control Information."

Click View Data... to open the Data Browser. This enables you to look at the data that is associated with the input link.

**Specifying Transaction Control Information**

If multiple input links are writing to a single ODBC data source, you can associate the links together as a transaction group. The transaction grouping feature is turned on and off by using the **Enable transaction grouping** check box on the Transaction Handling tab (it is off by default).

If transaction grouping is off, you can specify the following information on the Transaction Handling tab:

- Type a suitable value in the **Rows per transaction** field. This is the number of rows written before the data is committed to the data table. The default value is 0, that is, all of the rows are written before being committed to the data table.
- Type a suitable value in the **Parameter array size** field. This is the number of rows written at a time. The default is 1, that is, each row is written in a separate operation. If the current setting of **Parameter array size** causes available storage space to be exceeded at run time, you will be informed when you compile the job.

**Note:** If the **Parameter array size** setting conflicts with the **Rows per transaction** setting, the former takes precedence.

- Select a suitable **Isolation Level**. The isolation level specifies how potential conflicts between transactions (for example, dirty reads, nonrepeatable reads, and phantom reads) are handled.

If transaction grouping is enabled, the following rules govern the grouping of links:

- All of the input links in the transaction group must originate from the same Transformer stage.
- The ordering of the links within the transaction group is determined in the preceding Transformer stage.
- A transaction group cannot use a **Rows per transaction** or **Parameter array size** other than 1. Using an **Isolation level** of **Auto-commit** is permitted, but obviates the effect of organizing links in a transaction group.

You should be aware of the following facts about transaction groups (assuming that you commit on every row):

- A transaction starts at the beginning of each iteration of the Transformer stage that precedes the ODBC stage. Any uncommitted changes that are left over from a previous transaction are rolled back.
- The links in the transaction group are processed in the order laid down in the Transformer stage. Individual links might be skipped if the constraints laid down in the preceding Transformer stage so dictate.
- Each link in the transaction group can specify whether to roll back on failure. A rollback on any link causes the transaction to be abandoned and any subsequent links in the group to be skipped.
• Each link in the transaction group can be set to roll back if a constraint on that link is not met. Again, such a rollback causes the transaction to be abandoned and any subsequent links in the group to be skipped.

• The row counter for each link will be incremented only if the SQL that is associated with the link executes successfully and the transaction is successfully committed.

• The transaction ends after the last link in the transaction group is processed, unless a preceding link performs a rollback, in which case the transaction ends there.

To specify transaction control information for a transaction group:
1. Click the Transaction Handling tab.
2. Select the **Enable transaction grouping** check box. This option is only available if more than one link exists and the links come from the same Transformer stage. The current link must be one of those from the Transformer stage.
3. In the **Isolation Level** field, select an appropriate transaction isolation level. The isolation level specifies how potential conflicts between transactions (for example, dirty reads, nonrepeatable reads, and phantom reads) are handled. (If you select **Auto-commit**, you are specifying that every statement will effectively be executed in a separate transaction, which will obviate the advantages of transaction groups).
4. For transaction groups, **Rows per transaction** is automatically set to 1 and you cannot alter it.
5. For transaction groups, **Parameter array size** is automatically set to 1 and you cannot alter it.
6. Supply necessary details about the transaction group in the grid. The grid has a line for every link in the transaction group. The links are shown in transaction processing order, which is set in the preceding Transformer stage. Each line contains the following information:
   - **Input name**: The name of the input link. You cannot change this.
   - **On Skip**: This specifies whether to continue or roll back if a link is skipped due to a constraint on it not being satisfied. Choose **Continue** or **Rollback** from the list.
   - **On Fail**: This specifies whether to continue or roll back on failure of the SQL statement. Choose **Continue** or **Rollback** from the list.
   - **SQL**: Shows the SQL statement that is associated with the input link. You cannot change this, but clicking the cell will display the entire statement.

**Using a Generated Query**

**About this task**

You can write data to an SQL table by using an SQL statement that is constructed by the IBM InfoSphere DataStage. When you specify the table and the column definitions to use, the SQL statement is automatically constructed and can be viewed on the View SQL tab.

To use a generated query:

**Procedure**
1. Click the General tab on the Inputs page.
2. In the **Table name** field, select a table.
3. In the **Update action** field, specify how you want the data to be written. There are six options for a generated query:
   - **Clear the table, then insert rows**
   - **Insert rows without clearing**
   - **Insert new or update existing rows**
   - **Replace existing rows completely**
   - **Update existing rows only**
   - **Update existing or insert new rows**

   For a description of each update action, see the **Update action** section of the General tab.

4. Optional: In the **Description** field, type a description of the input link.
5. Click the Columns tab.
6. Edit the Columns grid to specify column definitions for the columns that you want to write.
   - The SQL statement is automatically constructed by using your chosen update action and the columns that you have specified.
7. Click the View SQL tab to view the SQL statement.
8. Click **OK** to close the ODBC stage editor. Changes are saved when you save your job design.

### Using a User-Defined SQL Statement

**About this task**

Instead of writing data to a table by using an SQL statement that is constructed by the IBM InfoSphere DataStage, you can specify your own SQL statement for each ODBC input link.

To specify an SQL statement:

**Procedure**

1. Click the General tab on the Inputs page.
2. In the **Update action** field, select **User-defined SQL**. The View SQL tab is replaced with the Enter SQL tab.
3. Click the Columns tab.
4. Edit the Columns grid to specify column definitions for the columns that you want to write.
5. Click the Enter SQL tab.
6. Type the SQL statement that you want to use. This statement must contain the table name, the type of update action that you want to perform, and the columns that you want to write.

   **Note:** You must also ensure that the statement contains the correct number of ? parameter markers. You must have a parameter marker for each column that you have defined on the Columns tab.
7. Click **OK** to close the ODBC stage editor. Changes are saved when you save your job design.
Using a Stored Procedure

About this task

Instead of writing data to a table by using an SQL statement, you can write data to a stored procedure. The columns that you define are bound (in order) to the input parameters in the stored procedure. The call syntax that is used to write the data is constructed by the IBM InfoSphere DataStage and can be viewed on the View SQL tab.

The procedure is called once for each row of data that is presented to the input link.

To use a stored procedure:

Procedure
1. Click the General tab on the Inputs page.
2. In the Update action field, select Call stored procedure. The Table name field is replaced by the Stored procedure name field.
3. In the Stored procedure name field, select the stored procedure.
4. Optional: In the Description field, type a description of the input link.
5. Click the Columns tab.
6. Edit the Columns grid to specify the column definitions. The column definitions are used as the input parameters to the stored procedure. You must have at least the same number of column definitions as the number of expected input parameters.
   The call statement is automatically constructed by using the stored procedure name and the columns that you have specified.
7. Click the View SQL tab to view the SQL statement.
8. Click OK to close the ODBC stage editor. Changes are saved when you save your job design.

Defining ODBC Output Data

When you extract data from an ODBC data source, the ODBC stage has an output link. The properties of this link and the column definitions of the data are defined on the Outputs page in the ODBC stage editor.

The Outputs page has the following field and up to six tabs. The tabs that are displayed depend on how you choose to specify the SQL statement to output the data:

Output name
This field displays the name of the output link. Choose the link that you want to edit from the list. The list contains all of the output links from the ODBC stage.

General
This tab is displayed by default and contains the following components:
- Table names. This field appears only when you select Generated query or User-defined SQL query. It contains the names of the tables or files that are being accessed. You can also use a job parameter to specify the table name.
• **Available tables.** This list appears only when you select **Generated query** or **User-defined SQL query.** It displays the names of the available tables or files that have definitions in the repository.

• **Add.** This button appears only when you select **Generated query** or **User-defined SQL query.** It adds a table from the **Available tables** list to the **Table names** field.

• **Stored procedure name.** This list is available only when you select **Stored procedure.** It displays the name of the stored procedure that you want to use. This list displays all of the stored procedure definitions in the **Table Definitions > StoredProcedures > DSN** folder in the repository.

• **Apply.** This button appears only when you select **Stored procedure.** It updates the Columns and Parameters tabs with the settings for the chosen stored procedure.

• **Generated query.** Specifies that the data is extracted by using an SQL statement that is constructed by InfoSphere DataStage. This is the default setting. When this option is selected, the Selection and View SQL tabs appear.

• **Stored procedure.** Specifies that the data is extracted by using a stored procedure. When this option is selected, the View SQL, Parameters, and Error Codes tabs appear.

• **User-defined SQL query.** Specifies that the data is extracted by using a user-defined SQL query. When this option is selected, the SQL Query tab appears.

• **Description.** Contains an optional description of the output link.

• **Browse...** Opens the Table Definitions window, allowing you to choose a suitable table or stored procedure definition.

**Columns**

This tab contains the column definitions for the data that is being output on the chosen link. It also specifies which columns are aggregated.

**Selection**

This tab appears when you select **Generated query.** It contains optional SQL SELECT clauses for the conditional extraction of data.

**Parameters**

This tab appears when you select **Stored procedure.** It contains the input parameters for a chosen stored procedure.

**View SQL**

This tab appears when you select **Generated query** or **Stored procedure.** It displays the SQL statement that is used to extract the data from the chosen table or tables. The SQL statement exists in two forms, depending on the type of input link that is required by the previous stage in the job:

• **SQL for reference inputs.** Displays the SQL statement that is used when this link is a reference input to a Transformer stage.

• **SQL for primary inputs.** Displays the SQL statement that is used in all other cases.

You cannot edit the SQL statement, but you can click **Copy** to copy it to the Clipboard for use elsewhere.

**Error Codes**

This tab appears when you select **Stored procedure.** It allows you to specify a space-separated list of values in the **Fatal errors** and **Warnings**
fields to handle raiserror calls within the stored procedure. You can also load predefined information from the repository by clicking **Load**.

**SQL Query**
This tab appears when you select **User-defined SQL query**. It contains a user-defined SQL query. This tab is divided into two areas:

- **SQL for primary inputs**. Contains a user-defined SQL query for a link that is a primary input to a Transformer stage, or an input to any other type of stage.
- **SQL for reference inputs**. Contains a user-defined SQL query for a link that is a reference input to a Transformer stage.

**Transaction Handling**
This tab allows you to specify a transaction isolation level for read data. The isolation level specifies how potential conflicts between transactions (for example, dirty reads, nonrepeatable reads, and phantom reads) are handled.

Click **View Data...** to open the Data Browser. This enables you to look at the data that is associated with the output link.

**Key Fields**
The column definitions for output links contain a key field. Key fields are used to join primary and reference inputs to a Transformer stage. For details about how to specify and use key fields, see *IBM InfoSphere DataStage Server Job Developer’s Guide*.

**Using a Generated Query**
When you select **Generated query**, data is extracted from an ODBC data source by using an SQL SELECT statement that is constructed by IBM InfoSphere DataStage. SQL SELECT statements have the following syntax:

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

After you specify the tables to use and the columns to be output from the ODBC stage, the SQL SELECT statement is automatically constructed and can be viewed by clicking the View SQL tab on the Outputs page.

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

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

The SELECT and FROM clauses are the minimum clauses that are required. These clauses are automatically generated by InfoSphere 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 that contain the selected columns.

- **WHERE clause**
  Specifies the criteria that rows must meet to be selected.
GROUP BY clause
Groups rows to summarize results. See “Aggregating Data” on page 98 for a description of how this clause is used.

HAVING clause
Specifies the criteria that grouped rows must meet to be selected. See “Aggregating Data” on page 98 for a description of how this clause is used.

ORDER BY clause
Sorts selected rows.

If you want to use the additional SQL SELECT clauses, you must specify them on the Selection tab on the Outputs page. The Selection tab is divided into two parts:
• 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 clause or an ORDER BY clause.

Using a WHERE Clause
You can use a WHERE clause to:
• Select only the data that meets certain criteria
• Join two tables from the same data source

To use a WHERE clause, type the column and the condition into the WHERE clause text entry box on the Selection tab on the Outputs page.

For example, if you have a table (Sales1) containing sales data, you can choose to only output data where the value in the Price column is greater than $10.00. In this case, type:
Price>10

Alternatively, if you are extracting data from two tables in the data source, you can use a WHERE clause to relate a column in one table to a column in the other table.

For example, Table1 contains the columns Pcode, OrderNo, and SaleDate and Table2 contains Pcode, CustNo, Quantity, and Cost. You can use the WHERE clause to join the two tables together by the related column. In this case, the column is Pcode and you type:
Table1.Pcode = Table2.Pcode

Note: Only one column definition called Pcode is loaded or inserted into the grid on the Columns tab.

You can also use a job parameter in the WHERE clause.

The SQL SELECT statement is automatically updated to include the WHERE clause. Click the View SQL tab to display the statement.

Using a HAVING Clause
If you use an ODBC stage to aggregate data, you can use a HAVING clause to specify conditions that the grouped data must meet before it is selected. For more information about using an ODBC stage to aggregate data, see “Aggregating Data” on page 98.
To use a HAVING clause, type the clause, column, and condition into the Other clauses text entry box on the Selection tab on the Outputs page.

For example, you could choose to only output summed quantities that are greater than or equal to 1000. In this case, type:

```
HAVING SUM(QtySold) >= 1000
```

You can also use a job parameter in the HAVING clause.

The SQL SELECT statement is updated automatically. Click the View SQL tab to display the statement.

**Using an ORDER BY Clause**

You can sort data based on a chosen column by including an ORDER BY clause in the SQL SELECT statement. Records are sorted by data in the chosen column before being output. You can specify a column name or a column position and whether to sort in ascending or descending order.

To use an ORDER BY clause, type the clause, column, and condition into the Other clauses text entry box on the Selection tab on the Outputs page.

For example, if your table contains a Name column, you might want to sort the column alphabetically (A to Z). In this case, type:

```
ORDER BY Name ASC
```

The SQL SELECT statement is updated automatically. Click the View SQL tab to display the statement.

**Aggregating Data**

If you are using a generated query, you can use an ODBC stage to aggregate data at the source instead of using an intermediate Aggregator stage. By aggregating data, you can add values in a particular column for all of the data records in a table. This summed total is then output from the stage.

You can aggregate data in two ways:
- Using an Aggregator stage
- Using an ODBC stage

If you aggregate data by using an ODBC stage, the columns to group by and sum together are also specified by the SQL SELECT statement. To specify the columns to group by and summarize, you must edit the column definitions in the Columns grid on the Columns tab.

For example, if you have a sales database (Sales1) it might contain the following columns: **Product**, **SaleDate**, and **QtySold**. If this database is updated daily, you have a record of how many of each product are sold each day. However, if you want to know how many of each product were sold since 01/01/96 you need to specify a WHERE clause for the **SaleDate** and group (and summarize) the data.

Because you want the total for each product, you need to group all of the occurrences of the same value in the **Product** column and sum the value in the **QtySold** column.
To group by a column, click the **Group** cell for the column definition that you want to group by and choose **Yes** from the list. In the example, you would choose the **Product** column to edit.

To summarize a column, edit the **Derivation** cell for the column that you want to aggregate (by using SUM or COUNT). By default the **Derivation** cell contains the name of the table and column in the format `tablename.columnname`. You can edit this cell to add SUM or COUNT. In the example, you would edit the **Derivation** cell for the **QtySold** column. The resulting expression would be `SUM(Sales1.QtySold)`.

You can use the Expression Substitution window to edit multiple **Derivation** cells at the same time. Select the columns and choose **Derivation Substitution...** from the pop-up menu.

When you group by or summarize columns, the SQL statement is automatically updated to include the GROUP BY clause and the aggregation expression. To view the SQL statement, click the View SQL tab on the Outputs page.

The SQL statement for this example would be:

```
SELECT Product, SUM(QtySold) FROM Sales1
  WHERE Saledate >= '01/01/96'
  GROUP BY Product;
```

See “[Using a WHERE Clause” on page 97](#) for details about how to use a WHERE clause.

### Using a User-Defined SQL Statement

**About this task**

Instead of using the SQL statement that is constructed by the IBM InfoSphere DataStage, you can specify your own SQL statement for each ODBC output link.

To specify an SQL statement:

**Procedure**

1. Click the General tab on the Outputs page.
2. Click **User-defined SQL query**. The SQL Query tab appears.
3. Click the SQL Query tab. When you first view this tab, the **SQL for primary inputs** and **SQL for reference inputs** fields might contain the SQL statements that are constructed by the InfoSphere DataStage. These statements are displayed if you selected **Generated query** or **Stored procedure** before you selected **User-defined SQL query**. You can modify or overwrite each statement to construct your own SQL query or call to a stored procedure.

The way you edit these fields depends on whether the output is a primary input to a stage or a reference input to a Transformer stage:

- If the output is a primary input to any stage, whether or not it is a Transformer stage, edit the **SQL for primary inputs** field. The SQL query must contain the same number of columns (and column names) as the SQL statement that is constructed by the InfoSphere DataStage.

  You must ensure that the table definitions for the output link are correct and represent the columns that are expected. The result set that is generated from this statement returns at least one row. If more than one result set is produced, only the first set is used.
Using a Stored Procedure

About this task

Instead of a user-defined SQL statement or a SQL statement that is constructed by the IBM InfoSphere DataStage, you can use a stored procedure to define the data that you want to extract for each ODBC output link.

You cannot use the output from a stored procedure as a reference input to a Transformer stage. If the ODBC output is an input to another stage in the job design, you must specify values for the stored procedure's parameters.

To use a stored procedure for an input:

Procedure

1. Click the General tab on the Outputs page.
2. Click Stored procedure. The Parameters, View SQL, and Error Codes tabs appear.
3. In the Stored procedure name list, select the stored procedure that you want to use. This list contains the names of the stored procedures that are defined in the Table Definitions > StoredProcedures > DSN folder in the repository.
   If you can’t see the name of the stored procedure that you want to use, you must define it in the repository. Alternatively, click Browse... to search the system for the stored procedure.
4. Click Apply. The Columns and Parameters tabs are updated with the column and parameter definitions for the chosen stored procedure.
5. Click the Parameters tab.
6. In the Value cell for each parameter, type suitable values. You can type constants or use job parameters.
   The call syntax that is used to extract the data is automatically updated with the parameter values. You can view this syntax on the View SQL tab.
7. Click OK to close the ODBC stage editor. Changes are saved when you save your job design.

Results

When the job runs, the stored procedure is called once with the given parameter values. The result set that is generated should contain at least one row.
Chapter 8. Building SQL statements

Use the graphical interface of SQL builder to construct SQL statements that run against databases.

You can construct the following types of SQL statements.

<table>
<thead>
<tr>
<th>SQL statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>Selects rows of data from a database table. The query can perform joins between multiple tables and aggregations of values in columns.</td>
</tr>
<tr>
<td>INSERT</td>
<td>Inserts rows in a database table.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Updates existing rows in a database table.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Deletes rows from a database table.</td>
</tr>
</tbody>
</table>

You can use the SQL from various connectivity stages that IBM InfoSphere DataStage supports.

Different databases have slightly different SQL syntax (particularly when it comes to more complex operations such as joins). The exact form of the SQL statements that the SQL builder produces depends on which stage you invoke it from.

You do not have to be an SQL expert to use the SQL builder, but it helps to have some familiarity with the basic structure of SQL statements in this documentation.

Starting SQL builder from a stage editor

You reach the SQL builder through the stage editors. Where the stage type supports the builder you will see a Build SQL button. Click this to open the SQL builder. For some stages you will have to select an appropriate access method for the button to be visible. See the documentation for the individual stage types for details.

The SQL builder is available to help you build select statements where you are using a stage to read a database (that is, a stage with an output link).

The SQL builder is available to help you build insert, update, and delete statements where you are using the stage to write to database (that is, a stage with an input link).

Starting SQL builder

Use the graphical interface of SQL builder to construct SQL queries that run against federated databases.

Procedure
1. In the Reference Provider pane, click Browse. The Browse Providers dialog box opens.
2. In the Select a Reference Provider type list, select Federation Server. In the Select a Federated Datasource tree, the list of database aliases opens.
3. Click a database alias. The list of schemas opens as nodes beneath each database alias.
4. In the SQL Type list, select the type of SQL query that you want to construct.
5. Click the SQL builder button. The SQL Builder - DB2 / UDB 8.2 window opens. In the Select Tables pane, the database alias appears as a node.

### Building SELECT statements

Build SELECT statements to query database tables and views.

**Procedure**

1. Click the Selection tab.
2. Drag any tables you want to include in your query from the repository tree to the canvas. You can drag multiple tables onto the canvas to enable you to specify complex queries such as joins. You must have previously placed the table definitions in the IBM InfoSphere DataStage repository. The easiest way to do this is to import the definitions directly from your relational database.
3. Specify the columns that you want to select from the table or tables on the column selection grid.
4. If you want to refine the selection you are performing, choose a predicate from the Predicate list in the filter panel. Then use the expression editor to specify the actual filter (the fields displayed depend on the predicate you choose). For example, use the Comparison predicate to specify that a column should match a particular value, or the Between predicate to specify that a column falls within a particular range. The filter appears as a WHERE clause in the finished query.
5. Click the Add button in the filter panel. The filter that you specify appears in the filter expression panel and is added to the SQL statement that you are building.
6. If you are joining multiple tables, and the automatic joins inserted by the SQL builder are not what is required, manually alter the joins.
7. If you want to group your results according to the values in certain columns, select the Group page. Select the Grouping check box in the column grouping and aggregation grid for the column or columns that you want to group the results by.
8. If you want to aggregate the values in the columns, you should also select the Group page. Select the aggregation that you want to perform on a column from the Aggregation drop-down list in the column grouping and aggregation grid.
9. Click on the Sql tab to view the finished query, and to resolve the columns generated by the SQL statement with the columns loaded on the stage (if necessary).

### Building INSERT statements

Build INSERT statements to insert rows in a database table.

**Procedure**

1. Click the Insert tab.
2. Drag the table you want to insert rows into from the repository tree to the canvas. You must have previously placed the table definitions in the IBM
3. Specify the columns that you want to insert on the column selection grid. You can drag selected columns from the table, double-click a column, or drag all columns.

4. For each column in the column selection grid, specify how values are derived. You can type a value or select a derivation method from the drop-down list.
   - **Job Parameters.** The Parameter dialog box appears. Select from the job parameters that are defined for this job.
   - **Lookup Columns.** The Lookup Columns dialog box appears. Select a column from the input columns to the stage that you are using the SQL builder in.
   - **Expression Editor.** The Expression Editor opens. Build an expression that derives the value.

5. Click on the **Sql** tab to view the finished query.

---

### Building UPDATE statements

Build UPDATE statements to update existing rows in a database table.

**Procedure**

1. Click the **Update** tab.
2. Drag the table whose rows you want to update from the repository tree to the canvas. You must have previously placed the table definitions in the IBM InfoSphere DataStage repository. The easiest way to do this is to import the definitions directly from your relational database.
3. Specify the columns that you want to update on the column selection grid. You can drag selected columns from the table, double-click a column, or drag all columns.
4. For each column in the column selection grid, specify how values are derived. You can type a value or select a derivation method from the drop-down list. Enclose strings in single quotation marks.
   - **Job Parameters.** The Parameter dialog box appears. Select from the job parameters that are defined for this job.
   - **Lookup Columns.** The Lookup Columns dialog box appears. Select a column from the input columns to the stage that you are using the SQL builder in.
   - **Expression Editor.** The Expression Editor opens. Build an expression that derives the value.
5. If you want to refine the update you are performing, choose a predicate from the **Predicate** list in the filter panel. Then use the expression editor to specify the actual filter (the fields displayed depend on the predicate you choose). For example, use the Comparison predicate to specify that a column should match a particular value, or the Between predicate to specify that a column falls within a particular range. The filter appears as a **WHERE** clause in the finished statement.
6. Click the **Add** button in the filter panel. The filter that you specify appears in the filter expression panel and is added to the update statement that you are building.
7. Click on the **Sql** tab to view the finished query.

---

### Building DELETE statements

Build DELETE statements to delete rows from a database table.
Procedure

1. Click the **Delete** tab.
2. Drag the table from which you want to delete rows from the repository tree to the canvas. You must have previously placed the table definitions in the IBM InfoSphere DataStage repository. The easiest way to do this is to import the definitions directly from your relational database.
3. You must choose an expression which defines the rows to be deleted. Choose a predicate from the **Predicate** list in the filter panel. Then use the expression editor to specify the actual filter (the fields displayed depend on the predicate you choose). For example, use the Comparison predicate to specify that a column should match a particular value, or the Between predicate to specify that a column falls within a particular range. The filter appears as a WHERE clause in the finished statement.
4. Click the **Add** button in the filter panel. The filter that you specify appears in the filter expression panel and is added to the update statement that you are building.
5. Click on the **Sql** tab to view the finished query.

The SQL builder Interface

The components in the top half of the SQL builder are common to all the types of statement you can build. The bottom half comprises a series of tabbed pages. What pages are available depends on the type of query you are building.

**Toolbar**

The SQL builder toolbar contains the following tools.

- **Clear Query** removes the field entries for the current SQL query.
- **Cut** removes items and placed them on the Microsoft Windows clipboard so they can be pasted elsewhere.
- **Copy** copies items and place them on the Windows clipboard so they can be pasted elsewhere.
- **Paste** pastes items from the Windows clipboard to certain places in the SQL builder.
- **SQL properties** opens the Properties dialog box.
- **Quoting** toggles quotation marks in table and column names in the generated SQL statements.
- **Validation** toggles the validation feature. Validation automatically occurs when you click OK to exit the SQL builder.
- **View Data** is available when you invoke the SQL builder from stages that support the viewing of data. It causes the calling stage to run the SQL as currently built and return the results for you to view.
- **Refresh** refreshes the contents of all the panels on the SQL builder.
- **Window View** allows you to select which panels are shown in the SQL builder window.
- **Help** opens the online help.

**Tree Panel**

This displays the table definitions that currently exist within the IBM InfoSphere DataStage repository. The easiest way to get a table definition into the repository is to import it directly from the database you want to query. You can do this via the Designer client, or you can do it directly from the shortcut menu in the tree panel.
You can also manually define a table definition from within the SQL builder by selecting **New Table...** from the tree panel shortcut menu.

To select a table to query, select it in the tree panel and drag it to the table selection canvas. A window appears in the canvas representing the table and listing all its individual columns.

A shortcut menu allows you to:
- Refresh the repository view
- Define a new table definition (the Table Definition dialog box opens)
- Import metadata directly from a data source (a sub menu offers a list of source types)
- Copy a table definition (you can paste it in the table selection canvas)
- View the properties of the table definition (the Table Definition dialog box opens)

You can also view the properties of a table definition by double-clicking on it in the repository tree.

**Table Selection Canvas**

Drag a table from the tree panel to the table selection canvas. If the desired table does not exist in the repository, you can import it from the database you are querying by choosing **Import Metadata** from the tree panel shortcut menu.

The table appears in a window on the canvas, with a list of the columns and their types. For insert, update, and delete statements you can only place one table on the canvas. For select queries you can place multiple tables on the canvas.

Wherever you try to place the table on the canvas, the first table you drag will always be placed in the top left hand corner. If you are building a select query, subsequent tables can be dragged before or after the initial, or on a new row underneath. Eligible areas are highlighted on the canvas as you drag the table, and you can only drop a table in one of the highlighted areas. When you place tables on the same row, the SQL builder will automatically join the tables (you can alter the join if it's not what you want).

When you place tables on a separate row, no join is added. An old-style Cartesian product of the table rows on the different rows is produced: `FROM FirstTable, SecondTable`.

For details about joining tables, see [Joining Tables](#).

Click the **Select All** button underneath the table title bar to select all the columns in the table. Alternatively you can double-click on or drag individual columns from the table to the grid in the **Select**, **Insert**, or **Update** page to use just those columns in your query.

With a table selected in the canvas, a shortcut menu allows you to:
- Add a related table (select queries only). A submenu shows you tables that have a foreign key relationship with the currently selected one. Select a table to insert it in the canvas, together with the join expression inferred by the foreign key relationship.
- Remove the selected table.
• Select all the columns in the table (so that you could, for example, drag them all to the column selection grid).
• Open a Select Table dialog box to allow you to bind an alternative table for the currently selected table (select queries only).
• Open the Table Properties dialog box for the currently selected table.

With a join selected in the canvas (select queries only), a shortcut menu allows you to:
• Open the Alternate Relation dialog box to specify that the join should be based on a different foreign key relationship.
• Open the Join Properties dialog box to modify the type of join and associated join expression.

From the canvas background, a shortcut menu allows you to:
• Refresh the view of the table selection canvas.
• Paste a table that you have copied from the tree panel.
• View data - this is available when you invoke the SQL builder from stages that support the viewing of data. It causes the calling stage to run the SQL as currently built and return the results for you to view.
• Open the Properties dialog box to view details of the SQL syntax that the SQL builder is currently building a query for.

**Selection Page**

The Selection page appears when you are using the SQL builder to define a Select statement. Use this page to specify details of your select query. It has the following components.

**Column Selection Grid**

This is where you specify which columns are to be included in your query. You can populate the grid in a number of ways:
• drag columns from the tables in the table selection canvas.
• choose columns from a drop-down list in the grid.
• double-click the column name in the table selection canvas.
• copy and paste from the table selection canvas.

The grid has the following fields:

**Column expression**

Identifies the column to be included in the query. You can specify:
• **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
• **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
• **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)
• **Lookup Column.** You can directly select a column from one of the tables in the table selection canvas.
Table
Identifies the table that the column belongs to. If you populate the column grid by
dragging, copying or double-clicking on a column from the table selection canvas,
the table name is filled in automatically. You can also choose a table from the
drop-down list.

To specify the table name at runtime, choose a job parameter from the drop-down
list.

Column Alias
This allows you to specify an alias for the column.

Output
This is selected to indicate that the column will be output by the query. This is
automatically selected when you add a column to the grid.

Sort
Choose Ascending or Descending to have the query sort the returned rows by the
value of this column. Selecting to sort results in an ORDER BY clause being added
to the query.

Sort Order
Allows you to specify the order in which rows are sorted if you are ordering by
more than one column.

Context Menu
A shortcut menu allows you to:
• Paste a column that you’ve copied from the table selection canvas.
• Insert a row in the grid.
• Show or hide the filter panel.
• Remove a row from the grid.

Filter Panel
The filter panel allows you to specify a WHERE clause for the SELECT statement
you are building. It comprises a predicate list and an expression editor panel, the
contents of which depends on the chosen predicate.

See Expression Editor for details on using the expression editor that the filter panel
provides.

Filter Expression Panel
This panel, at the bottom of the SQL builder window, displays any filters that you
have added to the query being built. You can edit the filter manually in this panel.
Alternatively you can type a filter straight in, without using the filter expression
editor.

Group Page
The Group page appears when you are using the SQL builder to define a select
statement. Use the Group page to specify that the results of a select query are
grouped by a column, or columns. Also, use it to aggregate the results in some of
the columns, for example, you could specify COUNT to count the number of rows
that contain a not-null value in a column.
The Group tab gives access to the toolbar, tree panel, and the table selection canvas, in exactly the same way as the Selection page.

### Grouping Grid

This is where you specify which columns are to be grouped by or aggregated on.

The grid is populated with the columns that you selected on the Selection page. You can change the selected columns or select new ones, which will be reflected in the selection your query makes.

The grid has the following fields:

- **Column expression.** Identifies the column to be included in the query. You can modify the selections from the Selection page, or build a column expression.
  - Job parameter. A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - Expression Editor. An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - Data flow variable. A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear).
  - Lookup Column. You can directly select a column from one of the tables in the table selection canvas.

- **Column Alias.** This allows you to specify an alias for the column. If you select an aggregation operation for a column, SQL builder will automatically insert an alias of the form Alison; you can edit this if required.

- **Output.** This is selected to indicate that the column will be output by the query. This is automatically selected when you add a column to the grid.

- **Distinct.** Select this check box if you want to add the DISTINCT qualifier to an aggregation. For example, a COUNT aggregation with the distinct qualifier will count the number of rows with distinct values in a field (as opposed to just the not-null values). For more information about the DISTINCT qualifier, see [SQL Properties Dialog Box](#).

- **Aggregation.** Allows you to select an aggregation function to apply to the column (note that this is mutually exclusive with the Group By option). See [Aggregation Functions](#) for details about the available functions.

- **Group By.** Select the check box to specify that query results should be grouped by the results in this column.

### Aggregation Functions

The aggregation functions available vary according to the stage you have opened the SQL builder from. The following are the basic ones supported by all SQL syntax variants.

The following aggregation functions are supported.

- **AVG.** Returns the mean average of the values in a column. For example, if you had six rows with a column containing a price, the six rows would be added together and divided by six to yield the mean average. If you specify the DISTINCT qualifier, only distinct values will be averaged; if the six rows only contained four distinct prices then these four would be added together and divided by four to produce a mean average.
- **COUNT.** Counts the number of rows that contain a not-null value in a column. If you specify the DISTINCT qualifier, only distinct values will be counted.
- **MAX.** Returns the maximum value that the rows hold in a particular column. The DISTINCT qualifier can be selected, but has no effect on this function.
- **MIN.** Returns the minimum value that the rows hold in a particular column. The DISTINCT qualifier can be selected, but has no effect on this function.
- **STDDEV.** Returns the standard deviation for a set of numbers.
- **VARIANCE.** Returns the variance for a set of numbers.

**Filter Panel**

The filter panel allows you to specify a HAVING clause for the SELECT statement you are building. It comprises a predicate list and an expression editor panel, the contents of which depends on the chosen predicate.

See [Expression Editor](#) for details on using the expression editor that the filter panel provides.

**Filter Expression Panel**

This panel displays any filters that you have added to the query being built. You can edit the filter manually in this panel. Alternatively you can type a filter straight in, without using the filter panel.

**Insert Page**

The Insert page appears when you are using the SQL builder to define an insert statement. Use this page to specify details of your insert statement. This page has the component **insert columns grid.**

**Insert Columns Grid**

This is where you specify which columns are to be included in your statement and what values they will take. The grid has the following fields:

**Insert Column**

Identifies the columns to be included in the statement. You can populate this in a number of ways:
- drag columns from the table in the table selection canvas.
- choose columns from a drop-down list in the grid.
- double-click the column name in the table selection canvas.
- copy and paste from the table selection canvas.

**Insert Value**

Identifies the values that you are setting the corresponding column to. You can specify one of the following in giving a value. You can also type a value directly into this field.
- **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
- **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
• **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)

• **Lookup Column.** You can directly select a column from one of the tables in the table selection canvas.

---

**Update Page**

The Update page appears when you are using the SQL builder to define an update statement. Use this page to specify details of your update statement. It has the following components.

**Update Column Grid**

This is where you specify which columns are to be included in your statement and what values they will take. The grid has the following fields:

**Update Column**

Identifies the columns to be included in the statement. You can populate this in a number of ways:

- drag columns from the table in the table selection canvas.
- choose columns from a drop-down list in the grid.
- double-click the column name in the table selection canvas.
- copy and paste from the table selection canvas.

**Update Value**

Identifies the values that you are setting the corresponding column to. You can specify one of the following in giving a value. You can also type a value directly into this field.

- **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
- **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
- **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)
- **Lookup Column.** You can directly select a column from one of the tables in the table selection canvas.

**Filter Panel**

The filter panel allows you to specify a WHERE clause for the update statement you are building. It comprises a predicate list and an expression editor panel, the contents of which depends on the chosen predicate.

See [Expression Editor](#) for details on using the expression editor that the filter panel provides.

**Filter Expression Panel**

This panel displays any filters that you have added to the query being built. You can edit the filter manually in this panel. Alternatively you can type a filter straight in, without using the filter panel.
The Delete page appears when you are using the SQL builder to define a delete statement. Use this page to specify details of your delete statement. It has the following components.

**Filter Panel**

The filter panel allows you to specify a WHERE clause for the delete statement you are building. It comprises a predicate list and an expression editor panel, the contents of which depends on the chosen predicate.

See "Expression Editor" for details on using the expression editor that the filter panel provides.

**Filter Expression Panel**

This panel displays any filters that you have added to the query being built. You can edit the filter manually in this panel. Alternatively you can type a filter straight in, without using the filter panel.

**Sql Page**

Click the Sql tab to view the generated statement. Using the shortcut menu, you can copy the statement for use in other environments.

For select queries, if the columns you have defined as output columns for your stage do not match the columns that the SQL statement is generating, use the Resolve columns grid to reconcile them. In most cases, the columns match.

**Resolve Columns Grid**

If the columns you have loaded onto your stage editor (the loaded columns) do not match the columns generated by the SQL statement (the result columns) you have defined, the Resolve columns grid gives you the opportunity to reconcile them. Ideally the columns should match (and in normal circumstances usually would). A mismatch would cause the metadata in your job to become out of step with the metadata as loaded from your source database (which could cause a problem if you are performing usage analysis based on that table).

If there is a mismatch, the grid displays a warning message. Click the Auto Match button to resolve the mismatch. You are offered the choice of matching by name, by order, or by both. When matching, the SQL builder seeks to alter the columns generated by the SQL statement to match the columns loaded onto the stage.

If you choose Name matching, and a column of the same name with a compatible data type is found, the SQL builder:

- Moves the result column to the equivalent position in the grid to the loaded column (this will change the position of the named column in the SQL).
- Modifies all the attributes of the result column to match those of the loaded column.

If you choose Order matching, the builder works through comparing each results column to the loaded column in the equivalent position. If a mismatch is found, and the data type of the two columns is compatible, the SQL builder:

- Changes the alias name of the result column to match the loaded column (provided the results set does not already include a column of that name).
• Modifies all the attributes of the result column to match those of the loaded column.

If you choose Both, the SQL builder applies Name matching and then Order matching.

If auto matching fails to reconcile the columns as described above, any mismatched results column that represents a single column in a table is overwritten with the details of the loaded column in the equivalent position.

When you click OK in the Sql tab, the SQL builder checks to see if the results columns match the loaded columns. If they don’t, a warning message is displayed allowing you to proceed or cancel. Proceeding causes the loaded columns to be merged with the results columns:
• Any matched columns are not affected.
• Any extra columns in the results columns are added to the loaded columns.
• Any columns in the loaded set that do not appear in the results set are removed.
• For columns that don’t match, if data types are compatible the loaded column is overwritten with the results column. If data types are not compatible, the existing loaded column is removed and replaced with the results column.

You can also edit the columns in the Results part of the grid in order to reconcile mismatches manually.

**Expression Editor**

The Expression Editor allows you to specify details of a WHERE clause that will be inserted in your select query or update or delete statement. You can also use it to specify WHERE clause for a join condition where you are joining multiple tables, or for a HAVING clause. A variant of the expression editor allows you to specify a calculation, function, or a case statement within an expression. The Expression Editor can be opened from various places in the SQL builder.

**Main Expression Editor**

To specify an expression:
• Choose the type of filter by choosing a predicate from the list.
• Fill in the information required by the Expression Editor fields that appear.
• Click the Add button to add the filter to the query you are building. This clears the expression editor so that you can add another filter if required.

The contents of the expression editor vary according to which predicate you have selected. The following predicates are available:
• **Between.** Allows you to specify that the value in a column should lay within a certain range.
• **Comparison.** Allows you to specify that the value in a column should be equal to, or greater than or less than, a certain value.
• **In.** Allows you to specify that the value in a column should match one of a list of values.
• **Like.** Allows you to specify that the value in a column should contain, start with, end with, or match a certain value.
• **Null.** Allows you to specify that a column should be null or should not be null.
Between
The expression editor when you have selected the Between predicate contains:

- **Column.** Choose the column on which you are filtering from the drop-down list. You can also specify:
  - **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)
  - **Column.** You can directly select a column from one of the tables in the table selection canvas.
- **Between/Not Between.** Choose Between or Not Between from the drop-down list to specify whether the value you are testing should be inside or outside your specified range.
- **Start of range.** Use this field to specify the start of your range. Click the menu button to the right of the field and specify details about the argument you are using to specify the start of the range, then specify the value itself in the field.
- **End of range.** Use this field to specify the end of your range. Click the menu button to the right of the field and specify details about the argument you are using to specify the end of the range, then specify the value itself in the field.

Comparison
The expression editor when you have selected the Comparison predicate contains:

- **Column.** Choose the column on which you are filtering from the drop-down list. You can specify one of the following in identifying a column:
  - **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)
  - **Column.** You can directly select a column from one of the tables in the table selection canvas.
- **Comparison operator.** Choose the comparison operator from the drop-down list. The available operators are:
  - = equals
  - <> not equal to
  - < less than
  - <= less than or equal to
  - > greater than
  - >= greater than or equal to
- **Comparison value.** Use this field to specify the value you are comparing to. Click the menu button to the right of the field and choose the data type for the value from the menu, then specify the value itself in the field.
In
The expression editor when you have selected the In predicate contains:

- **Column.** Choose the column on which you are filtering from the drop-down list. You can specify one of the following in identifying a column:
  - **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)
  - **Column.** You can directly select a column from one of the tables in the table selection canvas.

- **In/Not In.** Choose IN or NOT IN from the drop-down list to specify whether the value should be in the specified list or not in it.

- **Selection.** These fields allows you to specify the list used by the query. Use the menu button to the right of the single field to specify details about the argument you are using to specify a list item, then enter a value. Click the double right arrow to add the value to the list.
  To remove an item from the list, select it then click the double left arrow.

Like
The expression editor when you have selected the Like predicate is as follows. The fields it contains are:

- **Column.** Choose the column on which you are filtering from the drop-down list. You can specify one of the following in identifying a column:
  - **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear)
  - **Column.** You can directly select a column from one of the tables in the table selection canvas.

- **Like/Not Like.** Choose LIKE or NOT LIKE from the drop-down list to specify whether you are including or excluding a value in your comparison.

- **Like Operator.** Choose the type of Like or Not Like comparison you want to perform from the drop-down list. Available operators are:
  - **Match Exactly.** Your query will ask for an exact match to the value you specify.
  - **Starts With.** Your query will match rows that start with the value you specify.
  - **Ends With.** Your query will match rows that end with the value you specify.
  - **Contains.** Your query will match rows that contain the value you specify anywhere within them.

- **Like Value.** Specify the value that your LIKE predicate will attempt to match.
Null
The expression editor when you have selected the Null predicate is as follows. The fields it contains are:

- **Column.** Choose the column on which you are filtering from the drop-down list. You can specify one of the following in identifying a column:
  - **Job parameter.** A dialog box appears offering you a choice of available job parameters. This allows you to specify the value to be used in the query at run time (the stage you are using the SQL builder from must allow job parameters for this to appear).
  - **Expression.** An expression editor dialog box appears, allowing you to specify an expression that represents the value to be used in the query.
  - **Data flow variable.** A dialog box appears offering you a choice of available data flow variables (the stage you are using the SQL builder from must support data flow variables for this to appear).
  - **Column.** You can directly select a column from one of the tables in the table selection canvas.
- **Is Null/Is Not Null.** Choose whether your query will match a NULL or NOT NULL condition in the column.

Join
This predicate is only available when you are building an Oracle 8i query with an 'old style' join expression. The Expression Editor is as follows.

- **Left column.** Choose the column to be on the left of your join from the drop-down list.
- **Join type.** Choose the type of join from the drop-down list.
- **Right column.** Choose the column to be on the right of your query from the drop-down list.

Calculation/Function/Case Expression Editor
This version of the expression editor allows you to specify an expression within a WHERE or HAVING expression, or a join condition. Expression Editor dialogs are numbered to show how deeply you are nesting them. Fields in the Expression Editor panel vary according to the chosen predicate as follows:

Calculation
The expression editor when you have selected the Calculation predicate contains these fields:

- **Left Value.** Enter the argument you want on the left of your calculation. You can choose the type of argument by clicking the menu button on the right and choosing a type from the menu.
- **Calculation Operator.** Choose the operator for your calculation from the drop-down list.
- **Right Value.** Enter the argument you want on the right of your calculation. You can choose the type of argument by clicking the menu button on the right and choosing a type from the menu.

Functions
The expression editor when you have selected the Functions predicate contains these fields:

- **Function.** Choose a function from the drop-down list.
  The list of available functions depends on the database you are building the query for.
• **Description.** Gives a description of the function you have selected.

• **Parameters.** Enter the parameters required by the function you have selected.
  The parameters that are required vary according to the selected function.

**Case**

The case option on the expression editor enables you to include case statements in the SQL you are building. You can build case statements with the following syntax.

```
CASE WHEN condition THEN value
CASE WHEN...
ELSE value
```

or

```
CASE subject
WHEN match_value THEN value
WHEN...
ELSE value
```

The expression editor when you have selected the Case predicate contains these fields:

• **Case Expression.** This is the subject of the case statement. Specify this if you are using the second syntax described above (CASE subject WHEN). By default, the field offers a choice of the columns from the table or tables you have dragged to the table selection canvas. To choose an alternative, click the browse button next to the field. This gives you a choice of data types, or of specifying another expression, a function, or a job parameter.

• **When.** This allows you to specify a condition or match value for your case statement. By default, the field offers a choice of the columns from the table or tables you have dragged to the table selection canvas. To choose an alternative, click the browse button next to the field. This gives you a choice of data types, or of specifying another expression, a function, or a job parameter. You can access the main expression editor by choose case expression editor from the menu. This allows you to specify expressions such as comparisons. You would typically use this in the first syntax example. For example, you would specify grade=3 as the condition in the expression WHEN grade=3 THEN 'first class'.

• **Then.** Use this to specify the value part of the case expression. By default, the field offers a choice of the columns from the table or tables you have dragged to the table selection canvas. To choose an alternative, click the browse button next to the field. This gives you a choice of data types, or of specifying another expression, a function, or a job parameter.

• **Add.** Click this to add a case expression to the query. This clears the When and Then fields so that you can specify another case expression.

• **Else Expression.** Use this to specify the value for the optional ELSE part of the case expression.

**Expression Editor Menus**

A button appears to the right of many of the fields in the expression editor and related dialogs. Where it appears you can click it to open a menu that allows you to specify more details about an argument being given in an expression.

• **Bit.** Specifies that the argument is of type bit. The argument field offers a choice of 0 or 1 in a drop-down list.

• **Column.** Specifies that the argument is a column name. The argument field offer a choice of available columns in a drop-down list.
• **Date**. Specifies that the argument is a date. The SQL builder enters today’s date
  in the format expected by the database you are building the query for. You can
  edit this date as required or click the drop-down button and select from a
  calendar.

• **Date Time**. Specifies that the argument is a date time. The SQL builder inserts
  the current date and time in the format that the database the query is being built
  for expects. You can edit the date time as required.

• **Plaintext**. Allows you to select the default value of an argument (if one is
  defined).

• **Expression Editor**. You can specify a function or calculation expression as an
  argument of an expression. Selecting this causes the Calculation/Function
  version of the expression editor to open.

• **Function**. You can specify a function as an argument to an expression.
  Selecting this causes the Functions Form dialog box to open. The functions
  available depend on the database that the query you are building is intended
  for.

• **Job Parameter**. You can specify that the argument is a job parameter, the value
  for which is supplied when you actually run the IBM InfoSphere DataStage job.
  Selecting this opens the Parameters dialog box.

• **Integer**. Choose this to specify that the argument is of integer type.

• **String**. Select this to specify that the argument is of string type.

• **Time**. Specifies that the argument is the current local time. You can edit the
  value.

• **Timestamp**. Specifies that the argument is a timestamp. You can edit the value.
  The SQL builder inserts the current date and time in the format that the
  database that the query is being built for expects.

**Functions Form Dialog Box**
This dialog box allows you to select a function for use within an expression, and
specify parameters for the function.

The fields are as follows:

• **Function**. Choose a function from the drop-down list.
  The available functions depend on the database that you are building the query
  for.

• **Format**. Gives the format of the selected function as a guide.

• **Description**. Gives a description of the function you have selected.

• **Result**. Shows the actual function that will be included in the query as specified
  in this dialog box.

• **Parameters**. Enter the parameters required by the function you have selected.
  The parameters that are required vary according to the selected function.

**Function Dialog Box**:
This dialog box allows you to select a function for use within an expression, and
specify parameters for the function.

The fields are as follows:

• **Function**. Choose a function from the drop-down list.
  The available functions depend on the database that you are building the query
  for.
• **Format.** Gives the format of the selected function as a guide.
• **Description.** Gives a description of the function you have selected.
• **Result.** Shows the actual function that will be included in the query as specified in this dialog box.
• **Parameters.** Enter the parameters required by the function you have selected.
  The parameters that are required vary according to the selected function.

**Parameters Dialog Box**
This dialog box lists the job parameters that are currently defined for the job within which you are working. It also gives the data type of the parameter. Note that the SQL builder does not check that the type of parameter you are inserting matches the type expected by the argument you are using it for.

**Joining Tables**

When you use the SQL builder to help you build select queries, you can specify table joins within the query.

When you drag multiple tables onto the table selection canvas, the SQL builder attempts to create a join between the table added and the one already on the canvas to its left. If foreign key metadata is available for the tables, the SQL builder uses it. The join is represented by a line joining the columns the SQL builder has decided to join on. After the SQL builder automatically inserts a join, you can amend it.

When you add a table to the canvas, SQL builder determines how to join the table with tables that are on the canvas. The process depends on whether the added table is positioned to the right or left of the tables on the canvas.

To construct a join between the added table and the tables to its left:

1. SQL builder starts with the added table.
2. Determine if there is a foreign key between the added table and the subject table.
   - If a foreign key is present, continue to Step 3.
   - If a foreign key is not present, skip to Step 4.
3. Choose between alternatives for joining the tables that is based on the following precedence.
   - Relations that apply to the key fields of the added tables
   - Any other foreign key relation
   Construct an INNER JOIN between the two tables with the chosen relationship dictating the join criteria.
4. Take the subject as the next table to the left, and try again from step 2 until either a suitable join condition has been found or all tables, to the left, have been exhausted.
5. If no join condition is found among the tables, construct a default join.
   If the SQL grammar does not support a CROSS JOIN, an INNER JOIN is used with no join condition. Because this produces an invalid statement, you must set a suitable condition, either through the Join Properties dialog box, or by dragging columns between tables.
   An INNER JOIN is used with no join condition. Because this produces an invalid statement, you must set a suitable condition, either through the Join Properties dialog box, or by dragging columns between tables.
To construct a join between the added table and tables to its right:

1. SQL builder starts with the added table.
2. Determine if foreign key information exists between the added table and the subject table.
   - If a foreign key is present, continue to Step 3.
   - If a foreign key is not present, skip to Step 4.
3. Choose between alternatives based on the following precedence:
   - Relations that apply to the key fields of the added tables
   - Any other joins
   Construct an INNER JOIN between the two tables with the chosen relationship dictating the join criteria.
4. Take the subject as the next table to the right and try again from step 2.
5. If no join condition is found among the tables, construct a default join.
   - If the SQL grammar does not support a CROSS JOIN, an INNER JOIN is used with no join condition. Because this produces an invalid statement, you must set a suitable condition, either through the Join Properties dialog box, or by dragging columns between tables.
   - An INNER JOIN is used with no join condition. Because this produces an invalid statement, you must set a suitable condition, either through the Join Properties dialog box, or by dragging columns between tables.

**Specifying Joins**

There are three ways of altering the automatic join that the SQL builder inserts when you add more than one table to the table selection canvas:

- Using the Join Properties dialog box. Open this by selecting the link in the table selection canvas, right clicking and choosing **Properties** from the shortcut menu. This dialog allows you to choose a different type of join, choose alternative conditions for the join, or choose a natural join.

- Using the Alternate Relation dialog box. Open this by selecting the link in the table selection canvas, right clicking and choosing **Alternate Relation** from the shortcut menu. This dialog allows you to change foreign key relationships that have been specified for the joined tables.

- By dragging a column from one table to another column in any table to its right on the canvas. This replaces the existing automatic join and specifies an equijoin between the source and target column. If the join being replaced is currently specified as an inner or outer join, then the type is preserved, otherwise the new join will be an inner join.

Yet another approach is specify the join using a WHERE clause rather than an explicit join operation (although this is not recommended where your database supports explicit join statements). In this case you would:

1. Specify the join as a Cartesian product. (SQL builder does this automatically if it cannot determine the type of join required).
2. Specify a filter in the **Selection** tab filter panel. This specifies a WHERE clause that selects rows from within the Cartesian product.

If you are using the SQL builder to build Oracle 8i, Microsoft SQL Server, IBM Informix, or Sybase queries, you can use the Expression Editor to specify a join condition, which will be implemented as a WHERE statement. Oracle 8i does not support JOIN statements.
Join Properties Dialog Box

This dialog box allows you to change the type of an existing join and modify or specify the join condition.

The dialog box contains the following fields:

- **Cartesian product.** The Cartesian product is the result that is returned from two or more tables that are selected from, but not joined; that is, no join condition is specified. The output is all possible rows from all the tables selected from. For example, if you selected from two tables, the database would pair every row in the first table with every row in the second table. If each table had 6 rows, the Cartesian product would return 36 rows.

If the SQL builder cannot insert an explicit join based on available information, it will default to a Cartesian product that is formed with the CROSS JOIN syntax in the FROM clause of the resulting SQL statement: `FROM FirstTable CROSS JOIN SecondTable`. You can also specify a Cartesian product by selecting the Cartesian product option in the Join Properties dialog box. The cross join icon is shown on the join.

- **Table join.** Select the Table Join option to specify that your query will contain join condition for the two tables being joined. The Join Condition panel is enabled, allowing you to specify further details about the join.

- **Join Condition panel.** This shows the expression that the join condition will contain. You can enter or edit the expression manually or you can use the menu button to the right of the panel to specify a natural join, open the Expression Editor, or open the Alternate relation dialog box.

- **Include.** These fields allow you to specify that the join should be an outer join, where the result of the query should include the rows as specified by one of the following:
  - Select All rows from left table name to specify a left outer join
  - Select All rows from right table name to specify a right outer join
  - Select both All rows from left table name and All rows from right table name to specify a full outer join

- **Join Icon.** This tells you the type of join you have specified.

Alternate Relation Dialog Box

This dialog box displays all the foreign key relationships that have been defined between the target table and other tables that appear to the left of it in the table selection canvas. You can select the relationship that you want to appear as the join in your query by selecting it so that it appears in the list box, and clicking OK.

Properties Dialogs

Depending where you are in the SQL builder, choosing Properties from the shortcut menu opens a dialog box as follows:

- The Table Properties dialog box opens when you select a table in the table selection canvas and choose Properties from the shortcut menu.

- The SQL Properties dialog box opens when you select the Properties icon in the toolbox or Properties from the table selection canvas background.

- The Join Properties dialog box opens when you select a join in the table selection canvas and choose Properties from the shortcut menu. This dialog is described in Join Properties Dialog Box.
Table Properties Dialog Box

The Table Properties dialog box contains the following fields:

- **Table name.** The name of the table whose properties you are viewing.
  
  You can click the menu button and choose Job Parameter to open the Parameter dialog box (see Parameters Dialog Box). This allows you to specify a job parameter to replace the table name if required, but note that the SQL builder will always refer to this table using its alias.

- **Alias.** The alias that the SQL builder uses to refer to this table. You can edit the alias if required. If the table alias is used in the selection grid or filters, changing the alias in this dialog box will update the alias there.

SQL Properties Dialog Box

This dialog box gives you details about the SQL grammar that the SQL builder uses. It contains the following fields:

- **Description.** The name and version of the SQL grammar.
  
  The SQL grammar depends on the stage that you invoke the SQL builder from.

- **DISTINCT.** Specify whether the SQL builder supports the DISTINCT qualifier.
  
  If the stage supports it, the DISTINCT option is selected.
Chapter 9. Environment variables: ODBC connector

The ODBC Connector stage uses these environment variables.

**CC_GUARDIUM_EVENTS**

Set this environment variable to specify whether connectors report the InfoSphere DataStage context information to the InfoSphere Guardium Database Activity monitor.

When the value of this environment variable is set, the connectors report the InfoSphere DataStage context information such as host, project, job names, stage name and node ID that the stage is running on to the InfoSphere Guardium Database Activity monitor. When this environment variable is defined and set to any value, the connectors report context information to the Guardium server after the initial connection is established.

When this environment variable is undefined, the connectors do not attempt to report context information to Guardium servers. The setting of this environment variable applies to all database connectors in the job.

**CC_IGNORE_TIME_LENGTH_AND_SCALE**

Set this environment variable to change the behavior of the connector on the parallel canvas.

When this environment variable is set to 1, the connector running with the parallel engine ignores the specified length and scale for the timestamp column. For example, when the value of this environment variable is not set and if the length of the timestamp column is 26 and the scale is 6, the connector on the parallel canvas considers that the timestamp has a microsecond resolution. When the value of this environment variable is set to 1, the connector on the parallel canvas does not consider that the timestamp has a microsecond resolution unless the microseconds extended property is set even if the length of the timestamp column is 26 and the scale is 6.

**CC_MSG_LEVEL**

Set this environment variable to specify the minimum severity of the messages that the connector reports in the log file.

At the default value of 3, informational messages and messages of a higher severity are reported to the log file.

The following list contains the valid values:

- 1 - Trace
- 2 - Debug
- 3 - Informational
- 4 - Warning
- 5 - Error
- 6 - Fatal
**CC_ODBC_USE_TRUNCATE_FOR_TRUNCATE**

Set this environment variable to a nonzero value so that the connector generates a TRUNCATE TABLE command instead of the default DELETE FROM command.

When the value of this environment variable is set to a nonzero value, the **Table action** property is set to **Truncate**, and the **Generate truncate statement at runtime** property to **Yes** the connector generates a **TRUNCATE TABLE** command instead of the default **DELETE FROM** command.

**CC_ODBC_USE_NEW_DRIVER_MANAGER**

Set this environment variable to specify whether a DataDirect ODBC driver manager version 06.00.0127 (U0090) or later is used with InfoSphere Information Server, Version 8.5.

If you use the DataDirect ODBC driver manager version 06.00.0127 (U0090) or later, you must set this environment variable to 1.

This environment variable is not supported on Windows operating systems.

**CC_SE_TIMESTAMP_FF**

Set this environment variable to specify whether decimal point and fractional digits are included in the timestamp values, when the connector runs in server jobs.

When the environment variable is set to a value other than NONE, MICROSECONDS or SCALE, the behavior is the same as if the environment variable was not set. The environment variable values are case sensitive. When the environment variable is not set, the timestamp values that are produced by the job include a trailing decimal point and six fractional digits.

You can set the environment variable to the following values:

- **NONE**
  - The trailing decimal point and the fractional digits are both omitted.

- **MICROSECONDS**
  - The trailing decimal point and six fractional digits are included.

- **SCALE**
  - The trailing decimal point and $S$ fractional digits are included, where $S$ represents the value of the Scale attribute in the timestamp column definition. When the Scale attribute value is not defined for the column, the Scale attribute value of zero is assumed.

**CC_TRUNCATE_NSTRING_WITH_NULL**

Set this environment variable to truncate string data that includes the string 0x00.

When the value of this environment variable is set and when the input data contains a null character, the input data is truncated with 0x00 and the rest of the string is dropped.
CC_USE_EXTERNAL_SCHEMA_ON_MISMATCH

Set this environment variable to use an external schema rather than a design schema when the schemas do not match.

This schema is used for schema reconciliation. When the value of this environment variable is set, the behavior remains the same and is not changed from the old version.
Appendix A. Product accessibility

You can get information about the accessibility status of IBM products.

The IBM InfoSphere Information Server product modules and user interfaces are not fully accessible. The installation program installs the following product modules and components:

• IBM InfoSphere Business Glossary
• IBM InfoSphere Business Glossary Anywhere
• IBM InfoSphere DataStage
• IBM InfoSphere FastTrack
• IBM InfoSphere Information Analyzer
• IBM InfoSphere Information Services Director
• IBM InfoSphere Metadata Workbench
• IBM InfoSphere QualityStage

For information about the accessibility status of IBM products, see the IBM product accessibility information at http://www.ibm.com/able/product_accessibility/index.html.

Accessible documentation

Accessible documentation for InfoSphere Information Server products is provided in an information center. The information center presents the documentation in XHTML 1.0 format, which is viewable in most Web browsers. XHTML allows you to set display preferences in your browser. It also allows you to use screen readers and other assistive technologies to access the documentation.

The documentation that is in the information center is also provided in PDF files, which are not fully accessible.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.
Appendix B. Reading command-line syntax

This documentation uses special characters to define the command-line syntax.

The following special characters define the command-line syntax:

[ ] Identifies an optional argument. Arguments that are not enclosed in brackets are required.

... Indicates that you can specify multiple values for the previous argument.

| Indicates mutually exclusive information. You can use the argument to the left of the separator or the argument to the right of the separator. You cannot use both arguments in a single use of the command.

{} Delimits a set of mutually exclusive arguments when one of the arguments is required. If the arguments are optional, they are enclosed in brackets ([ ]).

Note:
- The maximum number of characters in an argument is 256.
- Enclose argument values that have embedded spaces with either single or double quotation marks.

For example:

**wsetsrc**[-S server] [-l label] [-n name] source

The source argument is the only required argument for the wsetsrc command. The brackets around the other arguments indicate that these arguments are optional.

**wlsac** [-l | -f format] [key...] profile

In this example, the -l and -f format arguments are mutually exclusive and optional. The profile argument is required. The key argument is optional. The ellipsis (...) that follows the key argument indicates that you can specify multiple key names.

**wrb** -import {rule_pack | rule_set}...

In this example, the rule_pack and rule_set arguments are mutually exclusive, but one of the arguments must be specified. Also, the ellipsis marks (...) indicate that you can specify multiple rule packs or rule sets.
Appendix C. How to read syntax diagrams

The following rules apply to the syntax diagrams that are used in this information:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol indicates that a syntax diagram is continued from the previous line.
  - The --->< symbol indicates the end of a syntax diagram.
- Required items appear on the horizontal line (the main path).

\[\text{required_item}\]

- Optional items appear below the main path.

\[\text{required_item} \rightarrow \text{optional_item}\]

If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

\[\text{required_item} \rightarrow \text{optional_item}\]

- If you can choose from two or more items, they appear vertically, in a stack.
  If you must choose one of the items, one item of the stack appears on the main path.

\[\text{required_item} \rightarrow \text{required_choice1} \rightarrow \text{required_choice2}\]

If choosing one of the items is optional, the entire stack appears below the main path.

\[\text{required_item} \rightarrow \text{optional_choice1} \rightarrow \text{optional_choice2}\]

If one of the items is the default, it appears above the main path, and the remaining choices are shown below.

\[\text{required_item} \rightarrow \text{default_choice} \rightarrow \text{optional_choice1} \rightarrow \text{optional_choice2}\]

- An arrow returning to the left, above the main line, indicates an item that can be repeated.
If the repeat arrow contains a comma, you must separate repeated items with a comma.

A repeat arrow above a stack indicates that you can repeat the items in the stack.

- Sometimes a diagram must be split into fragments. The syntax fragment is shown separately from the main syntax diagram, but the contents of the fragment should be read as if they are on the main path of the diagram.

**Fragment-name:**

- Keywords, and their minimum abbreviations if applicable, appear in uppercase. They must be spelled exactly as shown.
- Variables appear in all lowercase italic letters (for example, `column-name`). They represent user-supplied names or values.
- Separate keywords and parameters by at least one space if no intervening punctuation is shown in the diagram.
- Enter punctuation marks, parentheses, arithmetic operators, and other symbols, exactly as shown in the diagram.
- Footnotes are shown by a number in parentheses, for example (1).
Appendix D. Contacting IBM

You can contact IBM for customer support, software services, product information, and general information. You also can provide feedback to IBM about products and documentation.

The following table lists resources for customer support, software services, training, and product and solutions information.

Table 11. IBM resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description and location</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Support Portal</td>
<td>You can customize support information by choosing the products and the topics that interest you at <a href="http://www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server">www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server</a></td>
</tr>
<tr>
<td>Software services</td>
<td>You can find information about software, IT, and business consulting services, on the solutions site at <a href="http://www.ibm.com/businesssolutions/">www.ibm.com/businesssolutions/</a></td>
</tr>
<tr>
<td>My IBM</td>
<td>You can manage links to IBM Web sites and information that meet your specific technical support needs by creating an account on the My IBM site at <a href="http://www.ibm.com/account/">www.ibm.com/account/</a></td>
</tr>
<tr>
<td>Training and certification</td>
<td>You can learn about technical training and education services designed for individuals, companies, and public organizations to acquire, maintain, and optimize their IT skills at <a href="http://www.ibm.com/software/sw-training/">http://www.ibm.com/software/sw-training/</a></td>
</tr>
</tbody>
</table>
Appendix E. Accessing and providing feedback on the product documentation

Documentation is provided in a variety of locations and formats, including in help that is opened directly from the product client interfaces, in a suite-wide information center, and in PDF file books.

The information center is installed as a common service with IBM InfoSphere Information Server. The information center contains help for most of the product interfaces, as well as complete documentation for all the product modules in the suite. You can open the information center from the installed product or from a Web browser.

Accessing the information center

You can use the following methods to open the installed information center.

- Click the Help link in the upper right of the client interface.

  Note: From IBM InfoSphere FastTrack and IBM InfoSphere Information Server Manager, the main Help item opens a local help system. Choose Help > Open Info Center to open the full suite information center.

- Press the F1 key. The F1 key typically opens the topic that describes the current context of the client interface.

  Note: The F1 key does not work in Web clients.

- Use a Web browser to access the installed information center even when you are not logged in to the product. Enter the following address in a Web browser: http://host_name:port_number/infocenter/topic/com.ibm.swg.im.iis.productization.iisinfsv.home.doc/ic-homepage.html. The host_name is the name of the services tier computer where the information center is installed, and port_number is the port number for InfoSphere Information Server. The default port number is 9080. For example, on a Microsoft® Windows® Server computer named iisdocs2, the Web address is in the following format: http://iisdocs2:9080/infocenter/topic/com.ibm.swg.im.iis.productization.iisinfsv.nav.doc/dochome/iisinfsrv_home.html.

A subset of the information center is also available on the IBM Web site and periodically refreshed at http://pic.dhe.ibm.com/infocenter/iisinfsv/v9r1/index.jsp.

Obtaining PDF and hardcopy documentation

- A subset of the PDF file books are available through the InfoSphere Information Server software installer and the distribution media. The other PDF file books are available online and can be accessed from this support document: https://www.ibm.com/support/docview.wss?uid=swg27008803&wv=1

- You can also order IBM publications in hardcopy format online or through your local IBM representative. To order publications online, go to the IBM Publications Center at http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss
Providing comments on the documentation

Your feedback helps IBM to provide quality information. You can use any of the following methods to provide comments:

- To comment on the information center, click the Feedback link on the top right side of any topic in the information center.
- Send your comments by using the online readers’ comment form at [www.ibm.com/software/awdtools/rcf/](http://www.ibm.com/software/awdtools/rcf/)
- Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the information (if applicable). If you are commenting on specific text, include the location of the text (for example, a title, a table number, or a page number).
Notices and trademarks

This information was developed for products and services offered in the U.S.A.

Notices

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites.
sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to
IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

**Trademarks**

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

The following terms are trademarks or registered trademarks of other companies:

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The United States Postal Service owns the following trademarks: CASS, CASS Certified, DPV, LACS\textsuperscript{link}, ZIP, ZIP + 4, ZIP Code, Post Office, Postal Service, USPS and United States Postal Service. IBM Corporation is a non-exclusive DPV and LACS\textsuperscript{link} licensee of the United States Postal Service.

Other company, product or service names may be trademarks or service marks of others.
Index

A
Access to ODBD data sources configuring 9
adding deprecated stages to palette 6
Advanced tab
ODBC Enterprise stages 67, 72, 81
After SQL (node) property 44
After SQL property 44
aggregating data
using an ODBC stage 98
Array size property 44
Autocommit mode property 45

B
Before SQL (node) property 46
Before SQL property 45
Before/After SQL property 45

C
cannot find on palette 6
Case sensitive property 46
 CC_ODBC_USE_NEW_DRIVER_MANAGER environment variable 123
 CC_ODBC_USE_TRUNCATE_FOR_TRUNCATE environment variable 123
character set maps, defining
ODBC stages 87
Code page name property 47
Code page property 46
Column delimiter 47
Column name property 47
column values 47
Columns property 47
Columns tab
ODBC Enterprise stages 72, 81
ODBC stages 95
command-line syntax conventions 129
commands
syntax 129
configuration requirements 15
database drivers 15
connection parameters
ODBC stages 86
connector migration tool 1
Connector Migration Tool command line interface 4
Connector Migration Tool user interface 2
containers 1
migrating to use connectors 2, 4
Create statement property 48
CREATE TABLE statements 39
generated at design time 39
generated at runtime 40
customer support
contacting 133

d
Data
design time 32
viewing at design time 33
Data Browser 91
data connection objects
reusing in stage editor 28
Data Definition Language (DDL)
statements 38, 39, 40
data manipulation 33
Data Manipulation Language (DML)
statements 38, 41
data source connection
configuring connector access 27
reusing in stage editor 28
saving 28
testing connector access 27
Data source property 48
data sources
connection requirements 16
defining 16
supported 16
database connection
reusing in stage editor 28
installing 15
Drop statement property 49
Drop unmatched fields property 49
dsev script 11

E
Enable LOB references property 49
Enable partitioning property 50
Enable quoted identifiers property 50
End of data property 50
End of wave property 51
environment variables
ODBC connector 123
Error Codes tab (Outputs page)
ODBC stages 95

F
Fail on error property 51
Fail on row error 51
Fail on row error property 51
Fail on size mismatch property 52
Fail on type mismatch property 52

G
General tab
ODBC Enterprise stages 67, 68, 78, 81
ODBC stages 94
Generate create statement at runtime property 53
Generate drop statement at runtime property 53
Generate SQL property 53
Generate truncate statement at runtime property 53
generated queries
ODBC stages 92, 96

H
HAVING clauses
ODBC stages 97

design time services (continued)
validating SQL statements at design time 43
DML statements
DB2 connectors 41
DELETE statements 41
DELETE then INSERT statements 41
INSERT statements 41
INSERT then UPDATE statements 41
ODBC connectors 41
UPDATE statements 41
UPDATE then INSERT statements 41
Drop statement property 49
DROP TABLE statements 39
generated at design time 39
generated at runtime 40
Drop unmatched fields property 49

I
input links 17
ordering 20
ordering records 20
Input page
ODBC Enterprise stages 68
Inputs page
ODBC stages 89
Insert statement property 54
INSERT statements 33, 41
generated at run time 42
INSERT then UPDATE statements 41
installation requirements 15
Isolation level property 38, 54

J
job parameters
creating 28
removing from a property 29
selecting 29
jobs 1
execution order for the ODBC connector 17
migrating to use connectors 2, 4
using the ODBC connector 17

K
Key column property 55
key fields
ODBC stages 96

L
Large Objects (LOBs) 34, 35
legal notices 137
links
ordering 20
record processing 21
using with the ODBC connector 17
LOBs 34, 35
by reference 34
inline 34
transferring by reference using a connector 34
transferring inline using a connector 34
Log
column values 55
key values 55
Log multiple matches 56
Log multiple matches property 56
Logging
Column delimiter 55
column values 55
key values 55
lookup jobs
configuring 22
supported types 22

M
metadata 30
importing columns 30
metadata (continued)
importing tables 30
saving in stage editor 32
migrating to use connectors 1

N
NLS Map tab
ODBC Enterprise stages 68
NLS tab
ODBC stages 87
normal lookups 22
configuring 23
not on palette 6

O
ODBC connector
benefits 13
ODBC connectors 17
configuring a lookup operation 22
configuring as a source 19
configuring as a target 19
configuring data source connection 27
configuring in a normal lookup 23
configuring in a sparse lookup 24
Data Definition Language (DDL) statements 39
Data Manipulation Language (DML) statements 41
data source connection requirements 16
defining data source names 16
design-time data 32
importing metadata by columns 30
installing database drivers 15
jobs 17
metadata 30
reusing data source connection in stage editor 28
saving data source connection 28
saving metadata in stage editor 32
supported data sources 16
supported lookup types 22
testing data source connection 27
viewing design-time data 33
ODBC databases
setting up connectivity 9
ODBC Enterprise stages
Advanced tab 67, 72, 81
Columns tab 72, 81
General tab 67, 68, 78, 81
Input page 68
NLS Map tab 68
Output page 78
in Read Mode 78
with Odbcupsert for Reject Links 83
Partitioning tab 72, 78
Properties tab 68, 78, 81
ODBC stages
aggregating data 98
Columns tab
ODBC stages, Inputs page 90
Outputs page 95

P
palette
displaying stages 6
parallel engine
connecting to ODBC databases 9
parallel reads
partitioning data 37
parameters
creating for a job 28
removing from a property in a job 29
selecting for a job 29
Parameters tab
ODBC stages 95
partitioned data
parallel reads 37
Partitioning method property 56
Partitioning tab
ODBC Enterprise stages 72, 78
Password property 57
product accessibility
accessibility 127
product documentation
accessing 135
Properties tab
  ODBC Enterprise stages  68, 78, 81

Q
  queue manager connection
    reusing in stage editor  28

R
  Read After SQL (node)  57
  Read After SQL (node) statement from file property  57
  Read After SQL statement  57
  Read Before SQL (node)  58
  Read Before SQL (node) statement from file property  58
  Read Before SQL statement from file property  58
  Read select statement  58
  Read select statement from file property  58
  Record count property  59
  rules
    configuring for schema reconciliation  37
  runtime services
    data manipulation  33
    generating DDL statements at runtime  40
    generating SQL statements at runtime  42

S
  schema definitions
    importing in stage editor  30
    saving in stage editor  32
    schema reconciliation  35
    configuring rules  37
    rules  35
  Schema reconciliation property  59
  schemas
    design  35
    external  35
  Select statement property  59
  SELECT statements  33
    generated at run time  42
  Selection tab
    ODBC stages  95
  Session property  60
  setting environment variables for databases
    setting  10, 11
  software services
    contacting  133
  Sort order property  60
  sparse lookups  22
    configuring  24
  special characters
    in command-line syntax  129
  SQL (node)  57, 58
  SQL property  61
  SQL Query tab
    ODBC stages  96
  SQL Server data types
    in ODBC stages  87
  SQL statement  57, 58
  SQL statements  42
    syntax  96
    ways to write  38
    stage not on palette  6
    stages
      adding to palette  6
      stored procedures
        ODBC stages  94, 100
        support
          customer  133
          syntax  129
  Table action property  61
  Table name for Partitioning method property  61
  Table name for Table action property  62
  tabs  91
    Advanced  67, 72, 81
    Columns  72, 81
    Columns (Inputs page)  90
    Columns (Outputs page)  95
    Edit DDL  90
    Enter SQL  90
    Error Codes (Inputs page)  90
    Error Codes (Outputs page)  95
    General  67, 68, 78, 81, 94
    NLS  87
    NLS Map  68
    Parameters  95
    Partitioning  72, 78
    Properties  68, 78, 81
    Selection  95
    SQL Query  96
    Transaction Handling (Inputs page)  90
    Transaction Handling (Outputs page)  96
    View SQL (Inputs page)  90
    View SQL (Outputs page)  95
  Teradata connectors
    configuration
      as a source  19
      as a target  19
    configuring in a normal lookup  23
    configuring in a sparse lookup  24
    creating a lookup operation  22
    design-time data  32
    viewing design-time data  33
  transaction control information
    specifying  38
  Transaction property  62
  Truncate statement property  62
  TRUNCATE TABLE statements  39
    generated at design time  39
    generated at runtime  40

U
  Update statement property  62
  UPDATE statements  33, 41
    generated at run time  42
  UPDATE then INSERT statements  41
  user-defined SQL statements
    ODBC stages  93, 99
  Username property  63

V
  View SQL tab (Outputs page)
    ODBC stages  95

W
  web sites
    non-IBM  131
  WebSphere MQ connectors
    configuring as a source  19
    configuring as a target  19
    design-time data  32
    importing metadata columns in stage editor  30
    reusing data source connection in stage editor  28
    saving data source connection  28
    saving metadata in stage editor  32
    viewing design-time data  33
  WHERE clauses
    ODBC stages  97
  Write mode property  63