IBM InfoSphere DataStage and QualityStage
Version 11 Release 3

Connectivity Guide for IBM DB2 Databases

SC19-4256-00

IBM InfoSphere DataStage and QualityStage
Version 11 Release 3

Connectivity Guide for IBM DB2 Databases

SC19-4256-00

Note

Before using this information and the product that it supports, read the information in "Notices and trademarks" on page 159.

Contents

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 1. List of stages and corresponding connectors

Stage	Connector stage
DB2Z stage DB2 UDB API stage DB2 UDB Enterprise stage DB2 UDB Load stage	DB2 Connector
DRS Stage	DRS Connector
Java Client stage Java Transformer stage	Java Integration stage
Netezza Enterprise stage	Netezza Connector
ODBC Enterprise stage ODBC (Server) stage SQLServer Enterprise stage	ODBC Connector
Oracle OCI stage Oracle OCI Load stage Oracle Enterprise stage	Oracle Connector
Teradata API stage Teradata Enterprise stage Teradata Load stage Teradata Multiload stage	Teradata Connector
WebSphere® MQ stage	WebSphere MQ Connector

.

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 `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 **Start** > **Programs** > **IBM InfoSphere Information Server** > **Connector Migration Tool**.
2. In the Log on window, complete these fields:
   a. In the **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 **User name** field, enter your InfoSphere DataStage user name.
   c. In the **Password** field, enter your InfoSphere DataStage password.
   d. In the **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 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.
- **-b** *stage type*, where *stage type* is the built-in stage type to be migrated. This parameter is supported only on the command line, not on the user interface. Currently, only UniData 6 stages are supported. To migrate UniData 6 stages to UniData stages, specify `-b CUDT6Stage`.

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 were 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 were 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 was 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 the connector to use in place of the deprecated stages:

Table 2. Stages and corresponding connectors

Deprecated stage	Connector stage
DB2Z DB2 UDB API DB2 UDB Enterprise DB2 UDB Load	DB2 Connector
DRS	DRS Connector
Dynamic RDBMS	DB2 Connector Oracle Connector ODBC Connector

Table 2. Stages and corresponding connectors (continued)

Deprecated stage	Connector stage
Java Client Java Transformer	Java Integration stage
Netezza Enterprise	Netezza Connector
ODBC Enterprise	ODBC Connector
Oracle 7 Load Oracle OCI Load Oracle Enterprise	Oracle Connector
Teradata API Teradata Enterprise Teradata Load Teradata Multiload	Teradata Connector
WebSphere MQ	WebSphere MQ Connector

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 DB2 databases

To configure access to DB2 database, 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.

Configuring access to DB2 databases

To use the DB2 Connector stage in a job, you must configure DB2 environment variables and set the privileges for DB2 users. The DB2 connector connects to your databases by using the DB2 client on the InfoSphere DataStage nodes.

Before you begin

- Confirm that your system meets the system requirements for InfoSphere Information Server. Make sure that you are using a supported version of IBM DB2. For more information about system requirements, see http://www.ibm.com/software/data/infosphere/info-server/overview/.
- Install the IBM DB2 client on all InfoSphere DataStage nodes, and make sure that the client is working correctly.
- Use the DB2 Configuration Assistant to test the DB2 client and server connection. If the DB2 client fails to connect to the DB2 server, jobs that use the DB2 Connector stage also fail.
- Catalog each database in the DB2 client.
- InfoSphere DataStage runs many processes for each job. Ensure that DB2 resources, configuration parameters, and manager configuration parameters are configured properly.
- Make sure that the **DB2_PMAP_COMPATIBILITY** registry variable is set to `ON` to indicate that the distribution map size remains 4,096 (4-KB) entries. Though DB2® version 9.7 database for Linux, UNIX, and Windows supports distribution map entries up to 32,768 (32 KB), the DB2 connector supports only 4-KB entries in distribution maps.
- If you plan to use the DB2 connector with DB2 for z/OS in jobs with sparsely arriving data (such as jobs that use the Change Data Capture stage), ensure that the idle timeout value set in the DB2 **IDTHTION** subsystem parameter is longer than the longest expected interval of inactivity for the DB2 Connector stages in the job.

Procedure

1. Grant the InfoSphere DataStage users SELECT privileges on the following tables:

Table 3. Required SELECT privileges

DB2 product	Tables that require SELECT privileges
DB2 Database for Linux, UNIX, and Windows	SYSCAT.COLUMNS SYSCAT.KEYCOLUSE SYSIBM.SYSDBAUTH SYSCAT.TABLES

Table 3. Required SELECT privileges (continued)

DB2 product	Tables that require SELECT privileges
DB2 for z/OS	**Note:** Before the data is loaded to data to DB2 for z/OS, make sure the user has the GRANT ALL access on SYSIBM.SYSPRINT: SYSIBM.SYSCOLUMNS SYSIBM.SYSINDEXES SYSIBM.SYSKEYCOLUSE SYSIBM.SYSKEYS SYSIBM.SYSPRINT SYSIBM.SYSTABLESPACE SYSIBM.SYSTABLES SYSIBM.SYSTABLEPART SYSIBM.SYSUSERAUTH
DB2 Database for Linux, UNIX, and Windows and z/OS	SYSIBM.SYSDUMMY1 SYSIBM.SYSVIEWS

2. On DB2 for z/OS, ensure that the DBA runs the DSNTIJSG installation job to install the **DSNUTILS** stored procedure. The **DSNUTILS** stored procedure is required to start the bulk loader on DB2 for z/OS. For more information, see http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2z9.doc.inst/src/tpc/db2z_enabledb2supplstprocs.htm
3. Set the **DB2INSTANCE** environment variable to the instance in the DB2 client in which you cataloged the target database.

   You must set the **DB2INSTANCE** environment variable even if the stage accesses the default DB2 instance. The instance that is specified in the **DB2INSTANCE** environment variable becomes the default instance that is used by the connector. If you want to use a DB2 instance other than the default, then enter the name of that instance in the **Instance** property of the DB2 connector in the **Properties** tab. The DB2 client installs the default instances.

Table 4. Default instance installed by the DB2 client

Operating System	DB2 Instance
Linux or UNIX	`db2inst1`
Microsoft Windows	`DB2`

4. Add the path to the directory that contains the client libraries to the library path environment variable. The default path for client libraries is shown in the table.

Table 5. Default path for DB2 client libraries

Operating System	DB2 Instance
Linux or UNIX	`/opt/IBM/db2/V9/lib64`
Microsoft Windows	`C:\IBM\SQLLIB\BIN`

5. Optional: If the globalization map name for the DB2 connector job does not match the current system locale on the engine tier, then set the **DB2CODEPAGE** environment variable to a codepage corresponding to the map name. The DB2 code page can also be set by using a DB2 registry variable.

Configuring the DB2 native ODBC driver on AIX

To configure and use the DB2 ODBC driver on AIX operating systems, you must modify the DB2 environment variables and the **ODBCINI** environment variable.

Before you begin

- Confirm that your system meets the system requirements and that you are using a supported version of IBM DB2 database systems. For more information, see System Requirements.

Procedure

1. Update the 64-bit ODBC driver for DB2 on AIX operating systems so that the DataDirect driver manager can load the ODBC driver:
   a. Open the db2o.o driver file, which is in the *$DB2_HOME*/sqllib/lib64 directory.
   b. In the db2o.o driver file, add a link to the db2o.o.so file. For example, you can add the following link: `ln -s db2o.o db2o.o.so`
2. Add the following entry to the $ODBCINI file. The DataDirect driver manager uses the DriverUnicodeType=1 entry to work with the ODBC driver for DB2.

```
**************************************************
**************************************************
Driver=/home/db2inst1/sqllib/lib64/db2o.o.so
DriverUnicodeType=1
```

3. Set the **DB2INSTANCE** environment variable to the instance in the DB2 client in which you cataloged the target database.
4. Add the path to the directory that contains the client libraries to the library path environment variable.

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.

Operating system	Library path environment variable
Microsoft Windows	**PATH**
HP-UX	**SHLIB_PATH**
IBM AIX®	**LIBPATH**
Other supported Linux and UNIX operating systems, and HP-IA	**LD_LIBRARY_PATH**

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 DataStage administrator user (`dsadm` if you installed with the default name).
2. Back up the *IS_install_path*`/Server/DSEngine/dsenv` script. *IS_install_path* is the InfoSphere Information Server installation directory (`/opt/IBM/InformationServer` if you installed with the default path).
3. Open the `dsenv` script.
4. Add the path to the directory that contains the client libraries to the library path environment variable.
5. Set up your environment with the updated `dsenv` file.

   ```
   . ./dsenv
   ```
6. Restart the InfoSphere Information Server engine by entering the following commands:

   ```
   bin/uv –admin –stop
   bin/uv –admin –start
   ```
7. Assume root user privileges, directly with the **su** command or through the **sudo** command if the DataStage administrator user is part of the sudoers list.

   ```
   sudo su - root
   ```
8. Change to the ASB Agent home directory by entering the following commands:

   ```
   cd Install_directory/ASBNode/bin
   ```
9. Restart the ASB Agent processes by entering the following commands:

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

Results

After you restart the ASB Agent process, the InfoSphere Information Server services 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** system environment variable. 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. DB2 connector

You can use the DB2 connector to create jobs that read, write and load data. To use it you first need to configure it.

Configuring parallel processing for the DB2 connector with DB2 for Linux, UNIX, and Windows databases

You can configure the DB2 connector to read or write data in parallel mode. The distribution of parallel processes across hardware is based on the DB2 for Linux, UNIX, and Windows partition distribution.

Defining DB2 for Linux, UNIX, and Windows nodes and ETL nodes

You must define the DB2 for Linux, UNIX, and Windows nodes or ETL nodes in the parallel engine configuration file, also called the APT configuration file. The DB2 connector uses information specified in the APT configuration file to determine the number of processing nodes based on the number of partitions that are identified in the target table.

A DB2 connector can run on DB2 nodes or ETL nodes but not on both at the same time. If the APT configuration file specifies both DB2 and ETL nodes within the relevant node pool, the connector runs on DB2 nodes and ignores the ETL nodes. If no DB2 nodes are specified, the connector runs on ETL nodes. If the connector runs on ETL nodes, you must specify at least one ETL node.

If the connector runs on DB2 nodes, you must specify all of the DB2 nodes that the table spans. If a table has partitions on a DB2 node that is not listed in the configuration file, the job fails. The number of player processes for each DB2 node is dynamically determined based on the partition distribution of the selected table to ensure that one player process exists for each DB2 partition. Each player process processes data for the partitions that are hosted on that physical DB2 node.

You must specify one logical node for every physical node. If multiple logical nodes are defined for the same physical node, the connector uses the first logical node. For example, for a DB2 table that has 16 partitions and one physical ETL node, you set up the parallel engine configuration file with one logical node that maps to the single physical ETL node.

You can configure the DB2 connector to use a node pool or a node map constraint. If you do not specify a node map constraint, the connector uses the default node pool, which is identified in the APT configuration file with two double quotation marks (""). If the connector does not use the DB2 node pool, the connector uses the information that it receives from DB2 to locate the DB2 nodes in the configuration file. If the APT configuration file specifies the DB2 node pool, the DB2 connector does not automatically use it. To configure the connector to use the DB2 node pool, you must configure the node pool in the **Node pool and resource constraints** property on the Advanced page.

For example, if the connector runs on four DB2 nodes, each of which has four partitions, then four player processes are run on each node. If a table spans only 14 of the 16 partitions, then the first three nodes contain four players and the last

node contains two players. If the connector runs on ETL nodes, the connector distributes the player processes to ensure that each physical ETL node receives an equal load. The following table describes the distribution of parallel processes for a table with 16 partitions.

Table 6. Number of player processes that are running on each node

Total ETL nodes	Player processes on each node
1	16
2	8, 8
3	5, 5, 6
4	4, 4, 4, 4

When a job runs, the connector dynamically determines the number of player processes. If you run the job against a different table, you do not need to modify the parallel configuration file.

Limiting the number of processing nodes for bulk load

The DB2 connector uses one player process for each database partition. However, you can limit parallelism for bulk loads by configuring the DB2 connector to run one player process for multiple partitions.

For example, using many player processes on a table with many partitions affects performance adversely, set the **Limit parallelism** property to **Yes** to limit parallelism to one player process for multiple partitions.

When you limit the number of player processes, each player process handles multiple DB2 partitions. The connector partially partitions the data, and the DB2 server finishes the process by determining where to route the data for each partition. Because the DB2 server completes part of the process, the DB2 serve gets extra loaded. By adjusting the limit, you balance the load on the connector and the DB2 server.

To determine the optimal number of player processes to allocate, configure one player process for each DB2 partition, and then gradually reduce parallelism while you measure performance. If performance improves, continue to reduce parallelism until performance declines. If performance does not improve, keep the number of player processes at the default value, which is one player process for each DB2 partition.

Reducing parallelism improves performance only when the system is overloaded by the number of processes. If the system is powerful enough, the connector achieves the highest performance when it runs one player process for each DB2 partition.

The maximum number of processing nodes is equal to the number of DB2 partitions. If you specify a number greater than the number of DB2 partitions, the connector ignores the specified value and runs one player process for each DB2 partition. When the DB2 connector runs on DB2 nodes, the minimum number of player processes is equal to the number of physical DB2 nodes. You must specify one or more player processes for each physical DB2 node. When the DB2 connector runs on ETL nodes, the minimum number of player processes is **1** when the **Use direct connections property** is set to **No**. If the **Use direct connections property** is set to **Yes**, the minimum number of player processes is equal to the number of

physical DB2 nodes. When the DB2 connector runs on ETL nodes, if the player process limit is set to a value less than the number of ETL nodes, some of the nodes are not used. The DB2 connector requires one or more player processes for each DB2 node, so the connector fails if the minimum number of processing nodes is set too low.

Configuring the DB2 connector to use direct connections

If you configure the DB2 connector to use direct connections, the connector sends data directly to the appropriate DB2 data node without first sending the data to the DB2 admin node.

Before you begin

- Ensure that the DB2 connector runs on ETL nodes.
- Ensure that the DB2 server contains a separate admin node or the DB2 partitions reside on multiple DB2 data nodes.
- Ensure that all DB2 and ETL nodes are on the same network.

About this task

When the connector runs on ETL nodes, a DB2 client must be used to connect to a remote DB2 server. The DB2 server admin node is cataloged on the client, and all connections use the same cataloged entry. The DB2 connector routes data for all partitions through the admin node before sending it to the data nodes. When data is routed, additional processing and network activity is required which can overload the admin node and degrade performance.

To avoid this situation, you can configure the connector to send data directly to each partition without first passing through the admin node.

For example, consider the following table which shows how to catalog the DB2 nodes in an environment with one admin node and 16 data partitions spanning four data nodes.

Table 7. DB2 nodes in an environment with one admin node and 16 data partitions spanning four data nodes

DB2 nodes	Partitions	Description
admin	0	This node is an administration node. It stores system catalogs and other common information.
data1	1 2 3 4	This node is a data node with four partitions.
data2	5 6 7 8	This node is a data node with four partitions.
data3	9 10 11 12	This node is a data node with four partitions.
data4	13 14 15 16	This node is a data node with four partitions.

Procedure

1. To catalog an admin node and four physical DB2 nodes separately, specify the following commands:

```
CATALOG TCPIP NODE admin REMOTE admin SERVER 50000
CATALOG TCPIP NODE data1 REMOTE data1 SERVER 50000
CATALOG TCPIP NODE data2 REMOTE data2 SERVER 50000
CATALOG TCPIP NODE data3 REMOTE data3 SERVER 50000
CATALOG TCPIP NODE data4 REMOTE data4 SERVER 50000
```

2. In the `db2nodes.cgf` file, identify the partitions that are associated with each DB2 node.
3. Note the lowest partition number on each data node.
4. To catalog the remote database with appropriate database partitions, specify the command `CATALOG DB mydb AS mydb AT NODE name`. For example, if each of the four DB2 data nodes have four partitions, you issue the following commands:

   ```
   CATALOG DB mydb AS mydb AT NODE admin
   CATALOG DB mydb AS mydb1 AT NODE data1
   CATALOG DB mydb AS mydb5 AT NODE data2
   CATALOG DB mydb AS mydb9 AT NODE data3
   CATALOG DB mydb AS mydb13 AT NODE data4
   ```

   Where *mydb* is the name of the remote server.
5. Log in to the IBM InfoSphere DataStage and QualityStage Designer client.
6. Open the job to modify.
7. Double-click the DB2 Connector stage.
8. Set the **Use Direct Connections property** property to **Yes**.
9. Catalog each DB2 node separately on every ETL node.
10. Set the **Database** property to the name of the remote server. The connector automatically changes the value at run time by appending the appropriate suffix. Each DB2 data node must be cataloged with an alias that has a suffix which is the number of the lowest data partition on that data node.
11. Save and close the job.

High availability for DB2 systems

You can configure DB2 for high availability, to ensure that a failing DB2 node is replaced with a backup node.

To ensure that DB2 connector jobs are compatible with a DB2 high availability environment, complete the following steps:

- When a failing DB2 node is replaced by a backup node, the actual host name of a system can change. To support this situation, use virtual host names to catalog DB2 nodes.
- If you use the APT configuration file to specify DB2 nodes, then you must specify the actual DB2 host names in the APT configuration file for all active DB2 nodes and all backup DB2 nodes. The connector uses only the active DB2 nodes. When backup nodes become operational or active, the connector detects the change and looks for the backup nodes in the APT configuration file. If a DB2 node fails while a job runs, the job fails.
- You must restart the job after the backup nodes become operational or active.

Designing jobs by using the DB2 connector

You can use the IBM DB2 connector to develop jobs that read, write, and load data, and that store components from one job in the repository to reuse in another job.

Before you begin

- Configure access to DB2 databases.
- Verify if the user connecting to the DB2 database has the correct authority and privileges to perform the actions that your job requires.

Procedure

1. Import metadata from a DB2 source.
2. Access the DB2 Connector stage from the Designer client.
3. To set up the DB2 Connector stage to read data from a DB2 table:
   a. Configure the DB2 connector as a source of data.
   b. Set up column definitions.
   c. Define usage properties for reading data.
4. To set up the DB2 Connector stage to write data into a DB2 table:
   a. Configure the DB2 Connector stage as the target of data.
   b. Set up column definitions if they are not already specified for the link.
   c. Define usage properties for writing data.
   d. Optional: Define how data is partitioned.
5. Optional: Define how data is buffered.
6. Set up DB2 connector to look up data.
7. Create a reject link to manage rejected data.
8. Compile and run the job.

Importing DB2 metadata

Before you use the DB2 connector to read, write, or look up data, you can use InfoSphere Metadata Asset Manager to import the metadata that represents tables and views in a DB2 database. The imported metadata is then saved in the metadata repository.

Before you begin

- Confirm that your system meets the system requirements for InfoSphere Information Server. Make sure that you are using a supported version of IBM DB2. For more information about system requirements, see http://www.ibm.com/software/data/infosphere/info-server/overview/.
- Install the IBM DB2 client on all Engine Tier nodes, and make sure that the client is working correctly.
- Use the DB2 Configuration Assistant to test the DB2 client and server connection. If the DB2 client fails to connect to the DB2 server, metadata import also fails.
- Catalog each database you want to access in the DB2 client on each Engine Tier node.
- Make sure that you have the SELECT privilege for the following system schemas:

Table 8. Required SELECT privileges

DB2 product	Tables that require SELECT privileges
DB2 Database for Linux, UNIX, and Windows	SYSCAT.COLUMNS SYSCAT.INDEXCOLUSE SYSCAT.INDEXES SYSCAT.KEYCOLUSE SYSCAT.REFERENCES SYSIBM.SYSDBAUTH SYSCAT.TABCONST SYSCAT.TABLES
DB2 for z/OS	SYSIBM.SYSCOLUMNS SYSIBM.SYSFOREIGNKEYS SYSIBM.SYSINDEXES SYSIBM.SYSKEYCOLUSE SYSIBM.SYSKEYS SYSIBM.SYSRELS SYSIBM.SYSTABCONST SYSIBM.SYSTABLES

- Set the **DB2INSTANCE** environment variable to the instance in the DB2 client in which you cataloged the target database. You must set the **DB2INSTANCE** environment variable even if the stage accesses the default DB2 instance. The instance that is specified in the **DB2INSTANCE** environment variable becomes the default instance that is used by the connector. If you want to use a DB2 instance other than the default, then enter the name of that instance in the **Instance** property of the DB2 connector in the **Properties** tab. The DB2 client installs the default instances. The default DB2 instance on Linux or UNIX is `db2inst1` and on Microsoft Windows is `DB2`
- 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 follow:
  - On Linux or UNIX, the default path is `/opt/IBM/db2/V9/lib64`.
  - On Microsoft Windows, the default path is `C:\IBM\SQLLIB\BIN`.

About this task

By using the DB2 connector, you can import metadata about the following types of assets:

- The host computer that contains the DB2 database.
- The DB2 database.
- Database schemas.
- Database tables, system tables, and views. All imported tables are stored in the metadata repository as database tables.
- Database columns.
- Nicknames (The IBM InfoSphere Federation Server provides nickname support. To import nicknames, you must install and configure the IBM InfoSphere Federation Server).

Procedure

Import metadata by using InfoSphere Metadata Asset Manager. For more information about importing metadata by using InfoSphere Metadata Asset Manager, see the online product documentation in IBM Knowledge Center or the IBM InfoSphere Information Server Guide to Managing Common Metadata.

Defining a DB2 connector job

Use the InfoSphere DataStage and QualityStage Designer client to define a job by using the DB2 connector.

Procedure

1. From the Designer client, select **File** > **New** from the menu.
2. In the New window, select the **Parallel Job** or **Server Job** icon, and click **OK**.
3. On the left side of the Designer client in the Palette menu, select the **Database** category.
4. Locate **DB2** in the list of available databases, and click the down arrow to view the available stages.
5. Drag the DB2 Connector stage icon to the parallel or server canvas.
6. You can enter and modify the following attributes:
   - **Name of the DB2 Connector stage or link:** Modify the default name of the connector or the link. You can enter up to 255 characters. Alternatively, you can modify the name of the stage or link in the parallel canvas.
   - **Description:** Enter an optional description of the stage or link.
7. Click **Save**.

What to do next

Define properties to use DB2 connector as a source.

Reading data

To read data from a DB2 table by using the DB2 connector, you configure the DB2 connector to process data as a source. The DB2 connector runs an SQL statement and returns the results as a set of zero or more rows. In the source context, the connector extracts or reads data from an external DB2 data source.

The following figure shows an example of using the DB2 connector to read data. In this case, the DB2 connector reads data and then loads it from **DB2_Connector_1** into **DB2_Connector_2**. When you configure the DB2 connector to read data, you create only one output link, which is shown in the figure below transferring rows from **DB2_Connector_1** to **DB2_Connector_2**.

Figure 1. Example of reading data

Configuring the DB2 connector as a source

By configuring the IBM DB2 connector to process data as a source, you can use the DB2 Connector stage to read data.

Procedure

1. On the job design canvas, double-click the **DB2 Connector stage** icon.
2. Click the **Output** tab, then select the output link that you want to edit. By editing the output link you are setting up the DB2 Connector stage to be the source.
3. On the **Properties** tab in the **Connection** section, specify the instance, database, user name, and password that you want to use to make the connection.
4. In the **Alternate conductor settings** field, specify whether you want to provide alternate connection settings to use on the conductor node. The default value is **No**. If you select **Yes**, you must set the values for the subproperties, which are the instance, database, user name and password to use for the alternate connection.
5. If applicable, choose one of the following options for specifying the data connection:

Option	Description
Load	Loads an existing data connection from the repository.
Test	Tests the connection to your data source.

6. Click **Save** to save the connection settings that you specified.

Setting up column definitions

You set up column definitions for read operations and write operations in a similar way. You can also customize the columns grid, save column definitions for later use, and load predefined column definitions from the repository.

Procedure

1. On the job design canvas, double-click the **DB2 connector** icon.
2. Select either the **Output** tab or the **Input** tab depending on the type of link that you want to edit.
3. On the **Columns** tab, modify the columns grid to specify the metadata that you want to define.
   a. Right-click within the grid, and select **Properties** from the menu.
   b. In the Grid properties window, select the properties that you want to display and the order that you want them to be displayed. Then, click **OK**.
4. Enter column definitions for the table by using one of the following methods:

Option	Description
Method 1	1. In the **Column name** column, double-click inside the appropriate cell and type a column name. 2. For each cell in the row, double-click inside the cell and select the options that you want. 3. In the **Description** column, double-click inside the appropriate cell and type a description.

Option	Description
Method 2	1. Right-click within the grid, and select **Edit row** from the menu. 2. In the Edit column metadata window, enter the column metadata.

5. To share metadata between several columns, select the columns that you want to share metadata.
   a. Right-click and select **Propagate values**.
   b. In the Propagate column values window, select the properties that you want the selected columns to share.
6. To save the column definitions as a table definition in the repository, click **Save**.
   a. Enter the appropriate information in the Save Table Definition window, and then click **OK**.
   b. In the Save Table Definition As window, select the folder where you want to save the table definition, and then click **Save**.
7. To load column definitions from the repository, click **Load**.
   a. In the Table Definitions window, select the table definition that you want to load, and then click **OK**.
   b. In the Select Columns window, use the arrow buttons to move columns from the **Available columns** list to the **Selected columns** list. Click **OK**.

Defining properties for reading data

You must configure how the IBM DB2 connector operates in a job when reading data.

Before you begin

You must configure a database connection for the DB2 connector.

Procedure

1. On the job design canvas, double-click the **DB2 Connector stage** icon.
2. Click the **Output** tab, then select the output link that you want to edit.
3. Click the **Properties** tab.
4. On the **Properties** tab in the **Usage** section, specify how the connector operates in a job.
   a. Specify whether you want SQL statements generated at run time in the **Generate SQL** field.
   b. In the **Table** field, specify the table that you want to read.
   c. In the **Enable quoted identifiers** field, specify **Yes** to retain the case of all of the object names in DDL and DML statements. The default is **No**.
   d. In the **Before or After SQL** field, specify whether an SQL statement runs before or after data processing.
5. Click **OK** to save.

Configuring normal lookup operations

Data that is read by a database stage can serve as reference data to a Lookup stage. By default, this reference data is loaded into memory like any other reference link. When you perform a normal lookup operation, the DB2 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.

Procedure

1. Double-click on the DB2 Connector stage to open the link properties window.
2. From **Lookup Type** drop-down list, select **Normal**.
3. Click the **Columns** tab and define the columns that you want to use from the database to which the connector is connected.
4. Configure the properties on the **Properties** tab.
   a. Define and test your connection properties in the **Connection** section.
   b. In the **Usage** section, you can specify if you want auto-generated SQL statements or user-defined SQL statements. Specify **Key** as **Yes** or **No**.
      - If you specify **Generate SQL** as **Yes**, specify the **Table name** and then select the **Key** columns in the **Columns** tab in the lookup stage. To specify the **Key** columns drag the required columns from the primary link to the reference link. Note that this syntax means that many records are retrieved as opposed to the records that are retrieved in a sparse lookup operation.
      - If you specify **Generate SQL** as **No**, select the **Key** columns in the **Columns** tab in the lookup stage. To specify the **Key** columns drag the required columns from the primary link to the reference link. Specify the **Select statement** property. Type your SELECT statement in the **Select statement** property 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.
   c. Optional: Configure any other properties on the **Properties** tab.
5. Click **OK** to save the changes.
6. To map the input links to the output link, double-click the Lookup stage to open the stage editor.
   a. Drag or copy the columns from your input link to your output link to add the columns to the output link
   b. Define any conditions for a lookup failure by clicking the **Constraint** icon in the menu.
   c. 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. Click **OK**.
8. Save, compile, and run the job.

Configuring sparse lookup operations

Data that is read by a database stage can serve as reference data to a Lookup stage. By default, this reference data is loaded into memory like any other reference link. When directly connected as the reference link to a Lookup stage, you can configure the **Lookup Type** property of the DB2 connector to Sparse and send individual SQL statements to the database for each incoming Lookup row.

Before you begin

You must create a lookup operation job.

About this task

If the number of input rows to a stage is significantly smaller than the number of reference rows (1:100 or more) in a database table, you can configure the DB2 connector to perform a sparse lookup operation and send individual SQL statements to the database for each incoming lookup row. In the lookup operation job, 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 on the DB2 Connector stage to open the link properties window.
2. From **Lookup Type** list, select **Sparse**.
3. Click the **Columns** tab and define the columns that you want to use from the database to which the connector is connected.
4. Configure the properties on the **Properties** tab.
   a. Define and test your connection properties in the **Connection** section.
   b. In the **Usage** section, you can specify if you want auto-generated SQL statements or user-defined SQL statements. Specify **Generate SQL** as **Yes** or **No**.
      - If you specify **Generate SQL** as **Yes**, specify the **Table name** and the **Key** columns details in the **Columns** tab.
      - If you specify **Generate SQL** as **No**, specify 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 in the **Columns** tab, even if you do not use them in this lookup. You can then delete the fields in the columns that you do not need. The following syntax is an example of the first part of the SELECT statement: `select Field001,Field002,Field003`.
   c. Specify **Table name** in the **Properties** tab and then specify a WHERE clause to perform the lookup. 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, such as: `ORCHESTRATE.Field001`. The following SELECT statement is an example of the correct syntax of the WHERE clause: `select Field001,Field002,Field003 from MY_TABLE where ORCHESTRATE.Field001 = Field001`.
   d. Optional: Configure any other properties on the **Properties** tab.
5. Click **OK** to save the changes.
6. To map the input links to the output link, double-click the Lookup stage to open the stage editor.
   a. Drag or copy the columns from your input link to your output link to add the columns to the output link
   b. Define any conditions for a lookup failure by clicking the **Constraint** icon in the menu.
   c. 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. Click **OK**.

8. Save, compile, and run the job.

Writing data

To write data to a DB2 table by using the DB2connector, you configure the DB2 connector to process data as a target. The DB2 connector runs an SQL statement and returns the results as a set of zero or more rows. In the target context, the connector connects to the external DB2 data source and inserts, updates, or deletes data as required.

The following figure shows an example of using the DB2 connector to write data. In this case, the DB2 connector reads data from **DB2_Connector_1** and then loads the data into **DB2_Connector_2**. The DB2 connector then inserts, updates, or deletes data into **DB2_Connector_2** as required.

Figure 2. Example of writing data

Configuring the DB2 connector as a target

By configuring the DB2 connector to process data as a target, you can use the DB2 Connector stage to write data.

Procedure

1. On the parallel canvas, double-click the **DB2 Connector stage** icon.
2. Click the **Input** tab, then select the input link that you want to edit. By editing the input link you are configuring the DB2 Connector stage to be the target.
3. On the **Properties** tab in the **Connection** section, specify the connection settings for the DB2 connector.
4. If applicable, choose one of the following options for specifying the data connection:

Option	Description
Load	Loads an existing data connection from the repository.
Test	Tests the connection to your data source.

5. Click **Save** to save the connection settings that you specified.

Setting up column definitions

You set up column definitions for read operations and write operations in a similar way. You can also customize the columns grid, save column definitions for later use, and load predefined column definitions from the repository.

Procedure

1. On the job design canvas, double-click the **DB2 connector** icon.
2. Select either the **Output** tab or the **Input** tab depending on the type of link that you want to edit.

3. On the **Columns** tab, modify the columns grid to specify the metadata that you want to define.
   a. Right-click within the grid, and select **Properties** from the menu.
   b. In the Grid properties window, select the properties that you want to display and the order that you want them to be displayed. Then, click **OK**.
4. Enter column definitions for the table by using one of the following methods:

Option	Description
Method 1	1. In the **Column name** column, double-click inside the appropriate cell and type a column name. 2. For each cell in the row, double-click inside the cell and select the options that you want. 3. In the **Description** column, double-click inside the appropriate cell and type a description.
Method 2	1. Right-click within the grid, and select **Edit row** from the menu. 2. In the Edit column metadata window, enter the column metadata.

5. To share metadata between several columns, select the columns that you want to share metadata.
   a. Right-click and select **Propagate values**.
   b. In the Propagate column values window, select the properties that you want the selected columns to share.
6. To save the column definitions as a table definition in the repository, click **Save**.
   a. Enter the appropriate information in the Save Table Definition window, and then click **OK**.
   b. In the Save Table Definition As window, select the folder where you want to save the table definition, and then click **Save**.
7. To load column definitions from the repository, click **Load**.
   a. In the Table Definitions window, select the table definition that you want to load, and then click **OK**.
   b. In the Select Columns window, use the arrow buttons to move columns from the **Available columns** list to the **Selected columns** list. Click **OK**.

Defining properties for writing data

You must configure how the IBM DB2 connector operates in a job when writing data.

Before you begin

You must configure a database connection for the DB2 connector.

Procedure

1. On the job design canvas, double-click the **DB2 connector** icon.
2. Click the **Input** tab, then select the input link that you want to edit.
3. Click the **Properties** tab.
   a. Specify the **Write mode**.

b. In the **Table name** field, specify the name of the destination table that is used in the SQL statements that are meant for writing data. For the write mode, the table must exist. You can create the table at runtime using the **Create** or **Replace** table actions. The table name is used to generate Data Definition Language (DDL) statements. You must specify a table name if Write mode is set to **Bulk Load**, the **Generate SQL** property is set to **Yes**, or the **Table action** property is set to **Create**, **Drop**, or **Truncate**.

c. Specify whether you want SQL statements generated at run time in the **Generate SQL** field.

d. In the **Enable quoted identifiers** field, specify **Yes** to retain the case of all of the object names in DDL and DML statements. The default is **No**.

e. In the **SQL** field, specify the appropriate SQL statements.

f. In the **Table action** field, specify how you want tables to be created, or how you want rows to be edited or inserted in an existing destination table.

g. In the **Before or After SQL** field, specify whether an SQL statement runs before or after data processing.

4. Click **OK**.

Defining properties for bulk loading data

Use the bulk loading capabilities provided by the IBM DB2 connector to move large quantities of data into new or existing database tables efficiently. The bulk load operation can handle most data types, including large objects (LOBs) and user-defined types (UDTs).

Before you begin

- Verify that the user name in the connection properties for this stage has the correct database authority and privileges to use the bulk load properties that you specify. For more information, see the related topics that describe the DB2 **LOAD** command and **LOAD** authority.
- To load bulk data from a file, set up a File stage as input to the DB2 connector stage. The overall record size for the load is subject to limitations imposed by the parallel transaction engine. However, there is no limit when LOBs are passed by reference.

Procedure

1. On the job design canvas, double-click the **DB2 connector** icon.
2. In the top, left corner of the stage editor, select the input link.
3. On the **Properties** tab under the **Usage** category, in the **Write mode** field, select **Bulk load**.
4. In the **Table name** field, specify the name of the table into which the data will be loaded. If the schema name is different from the value specified for the user name, use a fully qualified table name in the format `schema.name`.
5. Specify appropriate values for the bulk load properties.
6. For loads that write to a DB2 partitioned database, expand the **Partitioned database configuration** category, and select **Yes**. Specify appropriate values for the partitioned database properties.
7. Click **OK**.

Partitioning data for a write operation

You can specify how the incoming data is partitioned or collected before the data is written to the IBM DB2 database. You can also specify that the data should be sorted before being written.

Before you begin

The availability of partition or collection type depends on whether you design your DB2 connector job to run in parallel or sequential mode.

- The **Partition type** list is available if you set the DB2 Connector stage to run in parallel mode. If you select a method from the list, the method overrides any current partitioning method.
- The **Collection type** list is available if you set the DB2 Connector stage to run in sequential mode, and the preceding stage is set to run in parallel mode.

Procedure

1. On the job design canvas, double-click the **DB2 Connector** icon.
2. In the top left corner of the stage editor, select the input link that you want to edit.
3. Click the **Partitioning** tab.
4. Select a partition type from the **Partition type/Collection type** list. Select **DB2 connector** if you want to apply the DB2 connector data partitioning or collection method to the data that you want to write. Click the **Properties** icon next to the **Partition type** list to specify the DB2 database, instance, and table.
5. Click **OK**.

Sorting data for a write operation:

The availability of the **Sort** option depends on the partitioning or collecting method that you select on the **Partitioning** tab in the stage editor. You cannot sort data if you select the **Auto** or **DB2 connector** partition or collection type.

Procedure

1. Select the **Sort** check box.
2. To retain previously sorted data sets, select one or both of the following options:
   - Select the **Stable** check box if you want to preserve previously sorted data sets.
   - Select both **Stable** and **Unique** as well if you want to retain only one of the multiple records that have identical sorting key values. In this case, the first record of the records that have identical sorting key values is retained.
3. Click **OK**.

Defining key columns for data sorting:

You can define which columns are key columns and which columns the data should be sorted on.

Procedure

1. Click any column in the **Available** list to move that column to the **Selected** list.
2. Click **OK**.

Defining sort direction, case sensitivity, and collating sequence:

In the stage editor, you can define the sort direction, case-sensitivity, and collating sequence of the data to be written.

Procedure

1. Right-click any of the columns that you moved to the **Selected** list.
2. From the shortcut menu, select the appropriate options. If national language support (NLS) is enabled you can specify the collation locale for the sort operation. The collation locale specifies precedence rules that are appropriate to the selected locale.
   a. Click the properties icon in the **Sort** area.
   b. In the **Sort Properties** window, choose a collation locale from the list.
3. Click **OK**.

Looking up data by using reference links

You can use the IBM DB2 connector to look up data directly from a DB2 table by using a reference link to link the DB2 Connector stage to a Lookup stage. The Lookup stage is a processing stage. It is used to perform lookup operations on a data set read into memory from any other Parallel job stage that can output data. You can specify a condition on each of the reference links, such that the stage will only perform a lookup on that reference link if the condition is satisfied.

About this task

A *reference link* represents a table lookup operation. You can use a reference link as an input link to a Lookup stage and as an output link from other types of stages, such as the DB2 Connector stage.

Procedure

1. On the job design canvas, drag a **DB2 connector** icon and a **Lookup stage** icon to the job design canvas. (The **Lookup stage** is located in the **Processing** category of the **Palette** menu.)
2. Join the stages by dragging a link from the DB2 Connector stage to the Lookup stage.
3. Right-click the link, and select **Convert to Reference** from the menu. The line changes to a dashed line to show that the link is a reference link.
4. Open the DB2 Connector stage editor by double-clicking the **DB2 connector** icon.
5. In the stage editor, define the database connection information for the stage, and then define the read operation for the reference link. When you are finished, click **OK**.
6. Open the Lookup stage editor by double-clicking the **Lookup stage** icon.
7. Define the lookup operation for the **Lookup stage**. The left pane of the editor shows the input links, and the right pane shows the output links. The metadata for these links is displayed below these panes. For each record of the source data set from the primary link, the Lookup stage performs a table lookup on each of the lookup tables attached by reference links. The table lookup is based on the values of a set of lookup key columns, one set for each table.
8. Click **OK**.

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.

Specifying job parameters

You can define parameters for an IBM DB2 connector job that allow you to set or change the value of a property at runtime. In the stage editor, you can create, select, or remove parameters from any properties that support parameters.

Procedure

1. On the job design canvas, double-click the **DB2 connector** icon.
2. In the top, left corner of the stage editor, select the link that you want to edit.
3. On the **Properties** tab click the property for which you want to create a job parameter. The **Use Job Parameter** button is displayed if the property supports parameters.
4. Create, select or remove a parameter as required.
5. Click **OK**.

Creating job parameters

Create a job parameter when you want to set or change the value of a property at runtime.

Procedure

1. On the job design canvas, double-click the **DB2 connector** icon.
2. In the top, left corner of the stage editor, select the link that you want to edit.
3. On the **Properties** tab click the property for which you want to create a job parameter. The **Use Job Parameter** button is displayed if the property supports parameters.
4. Click the **Use Job Parameter** button, and select **New Parameter**.
5. Specify values for the following fields:
   a. **Parameter Name:** Specify the name of the parameter. The value for this field cannot be a keyword in the BASIC programming language. This value is displayed in the **Properties** field with a number sign (#) at the beginning and end of the name.
   b. **Prompt**: Specify the prompt to display for this parameter.
   c. **Type**: Specify the type of the parameter. The default type corresponds to the property type for this parameter.
   d. Optional: **Default Value**: Specify the value to display for this parameter. You can modify this value at runtime.
   e. Optional: **Help Text**: Specify the help text to display for this parameter.
6. Click **OK**.

Example

In this example, you want to run the same job against different data sources. When you create the job, you define job parameters for the Data source, User name, and Password properties so that the connector can connect to the data source that you specify at runtime. When you run the job in InfoSphere DataStage and QualityStage Director, you can modify the values for these parameters in the Job Run Options window. The connector then uses the values that you specify to connect to the data source.

Selecting job parameters

Select an existing parameter when you want to reuse a parameter that is already defined for an IBM DB2 connector job.

Procedure

1. On the job design canvas, double-click the **DB2 connector** icon.
2. In the top left corner of the stage editor, select the link that you want to edit.
3. On the **Properties** tab, click the property for which you want to select a job parameter. If you can select a parameter for a property, you see the **Use Job Parameter** button.
4. Click the **Use Job Parameter** button, and select the parameter from the list.
5. Click **OK**.

Removing job parameters

Remove a parameter when it is no longer required in an IBM DB2 connector job.

Procedure

1. On the job design canvas, double-click the **DB2 connector** icon.
2. In the top left corner of the stage editor, select the link that you want to edit.
3. On the **Properties** tab, click the property for which you want to remove the job parameter.
4. Click the **Use Job Parameter** button, and click **Clear parameter**.
5. Click **OK**.

Defining data buffering

To improve performance and resolve bottlenecks, you can specify how input and output data for an IBM DB2 Connector stage is buffered. Although the size and operation of the buffer are usually the same for all links on all stages, you can modify the settings for specific links.

About this task

Important: By default, data is buffered so that no deadlocks occur. Be careful when changing data buffering settings because specifying inappropriate values might create a deadlock.

Any changes that you make to the properties on the **Advanced** tab are automatically reflected on the **Advanced** tab of the stage at the other end of the link.

To change the settings for data buffering:

Procedure

1. On the job design canvas, double-click the **DB2 connector** icon.
2. Select either the **Input link** tab or the **Output link** tab depending on the link that you want to edit.
3. On the **Advanced** tab, select the **Buffering mode** and specify the required values.
4. Click **OK** to save your changes and close the stage editor.

   For more details on the properties click **Help**.

Data type conversions

IBM InfoSphere DataStage supports a set of SQL data types that are different from DB2 SQL data types.

When you import metadata through the DB2 connector or read data by using the DB2 connector, the DB2 Connector stage converts the DB2 data types to InfoSphere DataStage data types. Conversely, when you write data to a target DB2 table through the DB2 connector, InfoSphere DataStage data types are converted to DB2 data types.

Data type conversions from DB2 to DataStage

When reading data, the DB2 Connector stage converts DB2 data types to InfoSphere DataStage data types.

Likewise, after metadata is imported through the DB2 connector, the DB2 data types are converted toIBM InfoSphere DataStage data types. The following table shows the mapping rules between DB2 data types and InfoSphere DataStage data types.

Table 9. DB2 data types and their equivalent InfoSphere DataStage data types

DB2 data types	InfoSphere DataStage data types
CHAR, 254	Char
VARCHAR, 32762	Varchar
LONGVARCHAR, 32700	LongVarChar
DECIMAL	Decimal
NUMERIC	Decimal
SMALLINT	SmallInt
INTEGER	Integer
BIGINT	BigInt
REAL	Real
FLOAT (same as DOUBLE)	Float
DOUBLE PRECISION	Double
DATE	Date
TIME	Time
TIMESTAMP	TimeStamp
CHAR FOR BIT DATA	Binary
VARCHAR FOR BIT DATA	VarBinary
LONGVARCHAR FOR BIT DATA	LongVarBinary
GRAPHIC, 127 double-byte character sets	NChar
VARGRAPHIC, 16336 double-byte character sets	NVarChar
LONGVARGRAPHIC, 16350 double-byte character sets	LongNVarChar
DECFLOAT	Decimal
XML	NVarChar
CLOB	LongVarChar

Table 9. DB2 data types and their equivalent InfoSphere DataStage data types (continued)

DB2 data types	InfoSphere DataStage data types
BLOB	LongVarBinary
DBCLOB	LongNVarchar

The XML data type can be imported as a LOB by selecting the **XML column as LOB** option during metadata import. XML columns can be represented as any of the string, ustring, or LOB data types.

Data type conversions from DataStage to DB2

When writing data, the DB2 Connector stage converts InfoSphere DataStage data types to DB2 data types.

In some cases, an exact conversion from an InfoSphere DataStage data type to a DB2 data type does not exist (for example, for the DB2 data types GRAPHIC and XML).

The following table shows the mapping rules between InfoSphere DataStage data types and DB2 data types.

Table 10. InfoSphere DataStage data types and their equivalent DB2 data types

InfoSphere DataStage data types	DB2 data types
SQL_BIGINT	BIGINT
SQL_BINARY	CHAR FOR BIT DATA
SQL_BIT	Unsupported
SQL_CHAR	CHAR
SQL_DATE	DATE
SQL_DECIMAL	DECIMAL
SQL_DOUBLE	DOUBLE PRECISION
SQL_FLOAT	FLOAT
SQL_INTEGER	INTEGER
SQL_LONGVARBINARY	LONG VARCHAR FOR BIT DATA
SQL_LONGVARCHAR	LONGVARCHAR
SQL_NUMERIC	DECIMAL
SQL_REAL	REAL
SQL_SMALLINT	SMALLINT
SQL_TIME	TIME
SQL_TIMESTAMP	TIMESTAMP
SQL_TINYINT	SMALLINT
SQL_VARBINARY	VARCHAR FOR BIT DATA
SQL_VARCHAR	VARCHAR
SQL_DECIMAL	DECFLOAT

Compiling and running DB2 connector jobs

You compile DB2 connector jobs into executable scripts that you can schedule and run.

Procedure

1. In the InfoSphere DataStage and QualityStage Designer client, open the job that you want to compile.
2. Click the **Compile** button.
3. If the Compilation Status area shows errors, edit the job to resolve the errors. After resolving the errors, click the **Re-compile** button.
4. When the job compiles successfully, click the **Run** button, and specify the job run options:
   a. Enter the job parameters as required.
   b. Click the **Validate** button to verify that the job will run successfully without actually extracting, converting, or writing data.
   c. Click the **Run** button to extract, convert, or write data.
5. To view the results of validating or running a job:
   a. In the Designer client, select **Tools** > **Run Director** to open the Director client.
   b. In the Status column, verify that the job was validated or completed successfully.
   c. If the job or validation fails, select **View** > **Log** to identify any runtime problems.
6. If the job has runtime problems, fix the problems, recompile, validate (optional), and run the job until it completes successfully.

DB2 connector properties for bulk loading to z/OS

Set the bulk load properties for the IBM DB2 connector to perform a bulk load on DB2 for z/OS®.

To enable the bulk load properties, you must set the **Write mode** to `Bulk load` and select **Bulk load to DB2 on z/OS** to `Yes`.

Load method
Specify the load method to use for loading input data into DB2 for z/OS.

If you select to bulk load to DB2 for z/OS, you must specify a load method. The following values are valid:

- MVS™ dataset(s)
- Batch pipe(s)
- USS pipe(s)

The connector runs in parallel for all the load methods. When the connector loads a partitioned table, a separate load utility is invoked for each partition.

Transfer
Set the transfer properties to transfer data to DB2 for z/OS. See "Transfer properties" on page 40 for more information.

DSN prefix
Specify a DSN prefix to use to construct the names of data sets, batch pipes, and USS pipes. This property is enabled for all load methods. If you

do not specify a DSN prefix, the value in the **Bulk load to DB2 on z/OS -> Transfer -> User** property is used. If a value is not specified for the **User** property, the value from the **Connection -> User name** property is used.

When you specify a DSN prefix, names are constructed in the following ways:

- Data files are named as prefix.IN.P#####, where ##### is the partition number, padded with zeros. If the table that is loaded is not partitioned, the partition number is 00000.
- Discard files are named as prefix.DSC.P#####, where ##### is the partition number, padded with zeros. If the table that is loaded is not partitioned, the partition number is 00000.
- Work files are named as prefix.WORK1.P##### and prefix.WORK2.P#####, where ##### is the partition number, padded with zeros. If the table that is loaded is not partitioned, the partition number is 00000.
- Error files are named as prefix.SYSERR.P#####, where ##### is the partition number, padded with zeros. If the table that is loaded is not partitioned, the partition number is 00000.
- Map files are named as prefix.SYSMAP.P#####, where ##### is the partition number, padded with zeros. If the table that is loaded is not partitioned, the partition number is 00000.

The DSN prefix includes date and time placeholders, which are replaced with the actual date and time when the job runs. The date format is *YYMMDD* The time format is *HHMMSS*. For example, if the value in the DSN prefix property is `USER1.`*DYYMMDD.THHMMSS* and the job is run on 8 February 2010 at 10:21:24 a.m on a nonpartitioned table by using the MVS dataset load method, the following names are used:

- USER1.D100208.T102124.IN.P00000
- USER1.D100208.T102124.DSC.P00000
- USER1.D100208.T102124.WORK1.P00000
- USER1.D100208.T102124.WORK2.P00000
- USER1.D100208.T102124.SYSERR.P00000
- USER1.D100208.T102124.SYSMAP.P00000

Batch pipe system ID

Provide the name for the batch pipe system.

This property is enabled only if you selected Batch pipes as the load method. This property determines how the data is transferred to DB2 on z/OS. If you specify a value for this property, then batch pipes are used to transfer the data to z/OS.

File(s) only

Use the Files only property to create data files on z/OS.

The Files only property is optional. Valid values for this property are Yes and No. If you select Yes, data files are created on z/OS but the LOAD utility is not invoked.

Note: This property is enabled only if the Load method is set to MVS dataset(s).

Device type

The Device type property is used to identify the device type to be used for data set allocation.

This property is optional. If you do not specify a value, SYSDA is the default value.

Partition number
Specify the partition to be loaded.

The Partition number property is optional. The value for this property must be an integer. If you do not specify a value, the data is loaded into all partitions.

Row count estimate
Use the Row count estimate property to provide the estimated number of rows to load into all the partitions combined.

This estimate is used to calculate the amount of disk space to allocate for the data sets. The Row count estimate property is optional. The value for this property must be an integer. If you do not specify a value, 1000 is the default value.

Statistics
Use the Statistics property to specify a display of statistics at the end of the load.

The Statistics property is optional. Valid values are None, All, Index, and Table. None is the default value.

Utility ID
Specify a name to use in DB2 to identify the execution of the load utility. The utility ID that you specify is appended with the suffix *.P#####*, where ##### is the partition number padded with zeros. If the table that is loaded is not partitioned, the partition number is 00000. If the length of the name is greater than 16 characters after the connector concatenates the utility ID and the suffix, the connector issues an error message. If no value is specified in the utility ID, the default value DB2ZLOAD.P##### is used.

This property is optional.

Load with logging
Use the Load with logging property to indicate whether logging must occur during the load process.

The Load with logging property is optional. Valid values are No and Yes. If you do not specify a value, No is used.

Set copy-pending
Use the Set copy-pending property to specify whether table space is set to copy-pending status.

The Set copy-pending property is optional. Valid values for this property are No and Yes. If you do not specify a value, No is the default value.

Note: This property is applicable when the value for Load with logging is No.

Encoding
Use the Encoding property to specify input data set encoding. See "Encoding properties" on page 40 properties for more information.

Image-copy function
Use the properties in this group to specify details about image copy and recovery files. Specify whether to run an Image-copy function after completing a bulk load job.

The valid values for this property are:

- Concurrent
- Full
- Incremental
- No (default)

You can configure the Image-copy function by specifying values to the properties that are displayed when you select a valid value.

Scope Specify the scope of the image-copy. The valid values are Full and Single partition.

Image-copy file
Specify the characteristics about the image-copy output file.

Image-copy backup file
Specify whether to create an image-copy file. The Image-copy backup file property is optional. The value for this property should be Boolean.

Recovery file
Specify whether to create the recovery file.

Recovery backup file
Specify whether to create an additional recovery file. The Recovery backup file property is optional.

Change limit percent 1
Specify the percentage limit of changed pages in the table space at which an incremental image-copy is to be taken. The value of this property can only be specified when Image copy is as same as the Incremental image-copy.

Change limit percent 2
Specify the percentage limit of changed pages in the table space at which an incremental image-copy is to be taken. The value of this property can only be specified when Image copy is as same as the Incremental image-copy, and when Change limit percent 1 is specified

Report only
The Report only file property is optional. Report only can be specified only when Image-copy is as same as the Incremental image-copy.

System pages
Specify whether the copy utility must put system pages at the beginning of the image-copy file.

Allow changes
Specify whether other programs can update the table space while the copy is running. This property is valid for the Image-copy function only.

Data file attributes
Set additional options to connect to DB2 connector to perform bulk load jobs. Set options, such as to create files on z/OS, to load a particular partition, and to display statistics at the end of the load. See “Data file attributes” on page 41 for more information.

Transfer properties

Set the transfer properties to transfer data to DB2 on IBM z/OS.

Transfer type
Specify the transfer type for the load operation. The only value is `FTP`.

Transfer to
Enter the name or IP address of the z/OS system where DB2 on IBM z/OS is running.

User Specify a user for the transfer operation. The **User** property is optional. The value for this property must be a string value. If the **User** property is not specified, the value of the **Connection** > **User name** property is used.

Password
Specify the password for the transfer operation. The value for this property must be a protected string value. The protected string value is a type of value you can select when creating or using a job parameter for a property. If the **Password** property is not specified, the value of the **Connection** > **Password** property is used.

Transfer command
Specify a command to run immediately before data transfer begins. For example, when using FTP, you can specify an FTP command like the following **QUOTE SITE** command `quote site vcount=7 datakeepalive=60`

Retry on connection failure
Select **Yes** to try to establish a connection again when the initial attempt to connect fails. Because the connector can initiate multiple connections to the transfer host simultaneously, the connector might exceed the maximum number of connections that the host can establish simultaneously. In this situation, multiple connection attempts might be necessary.

Number of retries
Enter the number of times to try to establish a connection after the initial attempt fails.

Interval between retries
Enter the time in seconds to wait between retries to establish a connection.

Note: While a DB2 connector attempts to establish a transfer connection, the DB2 connection process can timeout if the idle time exceeds the timeout that is specified by the **IDTHTOIN** DB2 subsystem parameter. Do not set the **Number of retries** and **Interval between retries** properties to values that, when multiplied together exceeds IDTHTOIN. Use these properties with caution.

Encoding properties

Use the Encoding properties to specify input data set encoding.

The Encoding property is optional. Valid values are EBCDIC, ASCII, UNICODE, and CCSID. If you do not specify a value, EBCDIC is used.

Character set
Specify the character set for data to be transferred to IBM z/OS. The default value is **ibm-1047-s390**.

CCSID

Use the CCSID property to specify the coded character set identifiers (CCSIDs) for the input file. The CCSID property is optional and the value must be an integer not enclosed by parentheses.

Data file attributes

Specify the name of the data set to use for the load utility. Specify additional options to connect to DB2 connector to perform bulk load jobs.

Input data files

Specify the properties for input data files for the load utility. If a data set name is specified, the data files are named as *value*.P#####, where *value* is the value in the data set name property and ##### is the partition number, padded with zeros. If a data set name is not specified, the data files are named based on the DSN prefix in the format prefix.IN.P#####. If the table that is loaded is not partitioned, the partition number is 00000.

Discard dataset properties

Specify the properties for discarding data sets for the load utility. If a data set name is specified, the discard files are named as *value*.P#####, where *value* is the value in the data set name property and ##### is the partition number, padded with zeros. If a data set name is not specified, the discard files are named based on the DSN prefix in the format prefix.DSC.P#####. If the table that is loaded is not partitioned, the partition number is 00000.

Error dataset properties

Specify the properties for error data set for the load utility. If a data set name is specified, the error files are named as *value*.P#####, where *value* is the value in the data set name property, and where ##### is the partition number, padded with zeros. If a data set name is not specified, the error files are named based on the DSN prefix in the format prefix.SYSERR.P#####. If the table that is loaded is not partitioned, the partition number is 00000.

Map dataset properties

Specify the properties for map data set for the load utility. If a data set name is specified, the map files are named as *value*.P#####, where *value* is the value in the data set name property, and where ##### is the partition number, padded with zeros. If a data set name is not specified, the map files are named based on the DSN prefix in the format prefix.SYSMAP.P#####. If the table that is loaded is not partitioned, the partition number is 00000.

Work1 dataset properties

Specify the properties for work1 data set for the load utility. This is a temporary dataset. If a data set name is specified, the temporary files are named as *value*.P#####, where *value* is the value in the data set name property, and where ##### is the partition number, padded with zeros. If a data set name is not specified, the temporary files are named based on the DSN prefix in the format prefix.WORK1.P#####. If the table that is loaded is not partitioned, the partition number is 00000.

Work2 dataset properties

Specify the properties for work2 data set properties for the load utility. This is a temporary dataset. If a data set name is specified, the temporary files are named as *value*.P#####, where *value* is the value in the data set name property, and where ##### is the partition number, padded with zeros. If a data set name is not specified, the temporary files are named

based on the DSN prefix in the format prefix.WORK2.P#####. If the table that is loaded is not partitioned, the partition number is 00000.

All the data files and datasets include date and time placeholders, which are replaced with the actual date and time when the job runs. The date format is *YYMMDD*. The time format is *HHMMSS*. For example, if the value in the data set name property is `USER1.`*DYYMMDD.THHMMSS* and the job is run on 8 February 2010 at 10:21:24 a.m on a nonpartitioned table the data set name is: USER1.D100208.T102124.P00000.

Data set properties

Assign the attributes to the data set properties to load data in to DB2 on z/OS.

Abnormal termination
Specify the action to take with the data set at abnormal job termination.

Data class
Specify the SMS data class (DATACLAS). This property is optional. The value for this property must be a string value.

Data set name
Specify a name for the Image-copy data set. This property is optional. The value for this property must be a string value.

Management class
Specify the SMS management class (MGMTCLAS). This property is optional. The value for this property must be a string value.

Normal termination
Specify the action to take with the data set at normal job termination.

Number of buffers
Specify the number of buffers. This property is optional. The value for this property must be an integer.

Primary allocation
Specify the z/OS disk space primary allocation amount. The range of values is from 1 to 1677215.

Secondary allocation
Specify the z/OS disk space secondary allocation amount. The range of values is from 1 to 1677215.

Space type
Specify the z/OS disk space allocation type. The Space type property is optional. Valid values are Cylinders and Tracks. The default value is Cylinders.

Status Specify the disposition status of the input data set used by load utility. This property is disabled when batch pipe system ID has any value. The default value is **Replace**. The valid values for this property are:

- Replace: Deletes an existing data set and creates a new data set.
- New: Indicates that the file does not currently exist.
- Old: Overwrites an existing data set or fails if the data set does not exist.
- Share: Identical to Old except that several jobs can read from the data set at the same time
- Append: Appends to the end of an existing data set or creates a new data set if it does not already exist.

Storage class
Specify the SMS storage class (STORCLAS). This property is optional. The value for this property must be a string value.

Unit Specify the device number, device type, or group name for the data set.

Volume(s)
Specifies the list of volume serial numbers for this allocation. The values can be entered with or without enclosing them in parentheses. This property is optional. The value for this property must be a string value or a list of string values that are separated by commas.

Troubleshooting

When running jobs using the DB2 connector, you might encounter errors that can be fixed by troubleshooting and adjusting values for properties or configurations.

If you encounter errors while running a job, ensure that rows are inserted correctly into your target tables. When processing rows, a DB2 database will reject all remaining rows to be processed after one of the following conditions occurs:

- A row cannot be inserted, and the value of the array size property is greater than 1
- The defined string length of the source data exceeds the defined length of its target column
- The source data contains a row with a character string that exceeds the length of the target column

Troubleshooting and adjusting values to enhance performance

Lower the array size property
To see detailed, row-level information about a failure, set the value of the array size property to 1. You modify this usage property on the **Properties** tab of the stage editor under the **Session** category. Setting the array size to 1 might affect performance.

Follow good development practices
To identify problems early in the development process, begin by defining the simplest possible job and confirm that the job runs successfully before adding complexity. Test the job frequently and do not add additional complexity until the job runs successfully.

Use SQL Builder to write SQL statements
To ensure that your SQL code is written properly, use the SQL Builder in the stage editor to build your SQL statements. If you have hand-coded your SQL statements and are experiencing problems, try using the SQL Builder to recreate the statements.

Avoid schema reconciliation problems
To avoid schema reconciliation problems, use the Import Connector Metadata wizard to import metadata. If the problems continue, modify the schema reconciliation usage properties. You modify these properties on the **Properties** tab of the stage editor under the **Session** category. Select **No** for **Fail on size mismatch** and **Fail on type mismatch**.

Use clean data
Use a Transformer stage to cleanse the data before sending it to the DB2 Connector stage. Adding a Transformer stage to your job might affect performance.

Use the same character set on the client and server

Make sure that the job and the DB2 database use the same character set. The character set that is used by a project or job might not match the character set for the schema that is defined in the DB2 database. For example, this problem can occur when the DB2 database uses the default character set for Linux, UTF-8, but the job uses the default character set for Microsoft Windows, Windows-1252. To resolve this problem, change the National Language Support (NLS) properties at the project or job level so that the project or job uses the same character set as the DB2 database. To modify NLS properties for a job, select **Edit** > **Job Properties**, and click the **NLS** tab.

The stage cannot obtain error messages from the server

Sometimes you are unable to obtain error messages when you view data or run a job.

Symptoms

You receive the error `Could not obtain the error message from server (function=SQLAllocHandle(SQL_HANDLE_ENV))` when you try to view data or run a job.

Resolving the problem

Verify that the instance property specifies a valid DB2 instance. If the Instance property is blank, verify that the *DB2INSTANCE* environment variable specifies a valid DB2 instance. More information on known issues with DB2 instances is available in the release notes.

Error loading connector library when you try to view data

Symptoms

When you try to view data, you receive the following error: `Error loading connector library`. The error message mentions symbols that are not exported in *libfcl*.

Resolving the problem

The *LIBPATH/LD_LIBRARY_PATH/PATH* environment variable that is used by the ASBAgent does not have directories listed in the correct order. The fcl library that is located in the ASBNode directory must be first in the *LIBPATH/LD_LIBRARY_PATH/PATH* environment variable that is used by the ASBAgent. Change your environment variable so that the ASBNode fcl library is found first. For example, the path must be similar to the following path: `LIBPATH=/opt/IBM/InformationServer/ASBNode/lib/cpp:....[add'l directories]`

Action String parameter is not valid or too long

You can encounter errors with the Action String parameter when the write mode of a job is set to "Bulk load".

Symptoms

When you run a job with a write mode that is set to “Bulk load”, you receive the following error: `SQL3009N The Action String parameter is not valid or too long`.

Resolving the problem

This error occurs because of a bug in DB2 9.5 FP2 and is documented in the following technote: http://www-01.ibm.com/support/docview.wss?rs=14

&uid=swg21322938. You must install DB2 9.5 FP3. After you install DB2 9.5 FP3, you must rebind your packages by following the directions in the following technote: http://www-01.ibm.com/support/docview.wss?uid=swg21449630

SQL0443N Routine error when you run a job

You can resolve some errors by rebinding packages.

Symptoms

You receive the following error message: `SQL0443N Routine "SYSIBM.SQLCOLUMNS" (specific name "COLUMNS") has returned an error SQLSTATE with diagnostic text "SYSIBM:CLI:-805". SQLSTATE=38553` when you run a job.

Resolving the problem

Rebind your packages. Follow the directions in the following technote: http://www-01.ibm.com/support/docview.wss?uid=swg21449630

Error when setting the active DB2 instance value

Symptoms

In the DB2 Connector stage GUI, when you change the Instance property and click the "Test Connection" or "View Data" buttons, you receive the following error: `An error occurred setting the active DB2 instance to value: <NewInstance>. The active DB2 instance is already set to value: <CurrentInstance>.`

Resolving the problem

Potential data loss and corruption from the connector

Symptoms

When you run a InfoSphere DataStage job, you see warnings similar to the following warning:

```
DB2: [IIS-CONN-DAAPI-000396] Writing the WCHAR column COL2 into a CHAR database column can cause data loss or corruption due to character set conversions.
```

Causes

WCHAR in the message refers to a wide character type, which is equivalent to NChar types.

The length of WCHAR column COL2 cannot be validated because the database column is CHAR and character set conversion is involved. Inadequate column lengths can lead to data truncation or unexpected errors.

This problem can happen for two reasons:

- The column type in the job is specified as NChar or NVarChar
- The column type in the job is specified as Char or VarChar with the extended attribute 'Unicode' set.

The column stores a Unicode string, but the CHAR database column stores data in a non-Unicode format. The **Unicode** field can have characters that cannot be converted into the database code page that is used for the CHAR column.

Even if all the characters can be converted, the connector cannot determine if the size of the database CHAR column is adequate to store every possible combination of Unicode characters that are contained in the job's Unicode column. The converted characters might occupy more than 1 byte.

Resolving the problem

Make sure that you use the same column type in both the job and the database. You must use type Char with type VarChar, and NChar with NVarChar. If the columns in the job are loaded from a saved table definition in DataStage, the columns automatically apply the extended attribute Unicode set. This makes Char and VarChar columns equivalent to NChar and NVarChar columns, which can also cause the warning message. Make sure that you manually remove the extended attribute.

Errors for jobs that access a remote DB2 instance

Symptoms

When you run a job that accesses a remote DB2 node and you specified an instance in the **Instance** property, you receive the following error: `An error occurred while getting the DB2 instance. Verify that a valid DB2 instance is specified in the Instance property. The method sqlegins returned reason code 0, SQLCODE -1,390.`

Resolving the problem

The instance that you specified in the **Instance** property might be invalid on the nodes that are defined in your configuration file. This problem often occurs when the connector is set to run on the DB2 nodes themselves. You can work around this issue by unsetting the *DB2INSTANCE* environment variable according to the following technote: http://www-01.ibm.com/support/docview.wss?uid=swg21381234

Jobs that fail are slow to report connection errors

Sometimes a job that uses the DB2 Connector can fail, and the failure is not immediately reported.

Symptoms

Jobs that fail are slow to report connection errors.

Causes

If the DB2 server fails when the DB2 Connector attempts to connect, the DB2 Connector might take hours to report the failure if the job hangs. This problem occurs when the DB2 server fails during the DB2 Connector's call to the `SQLConnect()` function.

Resolving the problem

This timeout is a DB2 feature. The timeout can be adjusted by setting the *ConnectTimeout* CLI/ODBC configuration keyword in the `db2cli.ini`, or by setting the *DB2TCP_CLIENT_CONTIMEOUT* registry variable on the DB2 client. The default timeout is to wait forever. The timeout can be set through the database alias. For information on the *ConnectTimeout* variable, see the following topic: http://pic.dhe.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.apdv.cli.doc/doc/r0021533.html

User-defined functions fail when the DB2 Connector queries the DB2 environment

Some user-defined functions can fail when they do not have privileges to access information about the DB2 environment

Symptoms

When you run a job with the DB2 Connector, you receive the following error: `"SQL0430N User defined function SYSPROC.ENV_GET_SYS_INFO" (specific name "ENV_GET_SYS_INFO") has abnormally terminated. SQLSTATE=38503 (CC_DB2Connection::queryServerHostName, file CC_DB2Connection.cpp, line 3,248)`

Causes

The DB2 Connector runs the following queries to obtain information about the DB2 environment:

- select HOST_NAME from SYSIBMADM.ENV_SYS_INFO
- select IS_INST_PARTITIONABLE from SYSIBMADM.ENV_INST_INFO

These queries require the following items:

- SELECT or CONTROL privilege on the ENV_SYS_INFO administrative view and EXECUTE privilege on the ENV_GET_SYS_INFO table function
- SELECT or CONTROL privilege on the ENV_INST_INFO administrative view and EXECUTE privilege on the ENV_GET_INST_INFO table function

Resolving the problem

Add these privileges to the user that is specified in the DB2 Connector's *User name* property, and run the job again.

The DB2 Connector cannot find any available nodes in the APT configuration file

When the DB2 connector cannot find available notes in the APT configuration file, you must add the DB2 node pool in that APT file.

Symptoms

When you run a job, you receive the following error: `The connector could not find any available nodes in the APT configuration file. This usually occurs when a node pool constraint is specified for a connector stage, but it is not defined in the APT configuration file (CC_DB2Configuration::validateEnvironment, file CC_DB2NodeNegotiation.cpp, line 787`

Causes

This error occurs when a node pool constraint is specified for a DB2 Connector stage, but the node pool is missing from the APT configuration file. This problem occurs after you migrate a job that uses the IBM DB2/UDB Enterprise stage to use the DB2 Connector stage with the Connectivity Migration tool.

The IBM DB2/UDB Enterprise stage requires a node pool that is called *DB2* is defined. When jobs are migrated, the DB2 Connector stage in those jobs has a *DB2* node pool constraint specified. Edit your APT file and remove the "DB2" node pool before you run your migrated jobs, you encounter this error.

Resolving the problem

Add the *DB2* node pool back into the APT file, and then edit the node pool constraints on the DB2 Connector stages.

Errors when you set the active DB2 instance value

An error occurred setting the active DB2 instance to value `NewInstance`. The active DB2 instance is already set to value `CurrentInstance`.

Symptoms
The error message indicates that the DB2 connector cannot set the active DB2 instance to the specified value.

Causes
This error happens because each process that uses the DB2 connector can be associated with only one DB2 instance. The error occurs when two or more DB2 connector stages are in a job, and a different instance is specified in each stage. The error can also occur when the **Instance** property is changed when you view data or import metadata. The DB2 connector can test connections, view data, and import metadata from only one DB2 instance. A single continuously running ASBAgent process handles all of the activity that happens at design time. The first DB2 instance that is used to test a connection, view data, or import metadata, is the instance that the connector uses for any design-time actions.

Resolving the problem
To use a different DB2 instance, complete one of the following actions:

- Restart the ASBAgent service on Microsoft® Windows or the daemon on UNIX.
- Enable Connector Access Service multiple instance support.

To enable the Connector Access Service multiple instance support, complete the following steps:

1. Stop the ASBAgent and ensure that the `ASBAgent.exe` process is shut down.
2. Specify the free socket port and the port number range by adding the following lines to `ASBNode/conf/cas.properties`. The port number range depends on your environment.

   ```
   multipleprocess.enable=1
   multipleprocess.port.range.min=30000
   multipleprocess.port.range.max=30010
   ```
3. Restart the ASBAgent service.

Timeout error when the connector waits for the LOAD utility to begin reading data
The FTP server timed out waiting for the LOAD utility to begin reading data. A timeout can occur when the FTP server is configured with a short FIFOOPENTIME or when the LOAD utility starts slowly.

Symptoms
You receive the following error while the connector waits for the LOAD utility to begin reading data:

```
Transfer to dataset dataset failed with error: 550 DELE fails:
dataset does not exist.
...
125-Waiting for read process to open /tmp/dataset
125 Transfer request aborted450 timer expired waiting for read process to
open /tmp/dataset
450 timer expired waiting for read process to open /tmp/dataset
```

The *dataset* variable is the name of the data set that is generated from the data set names that are specified in the DB2 connector.

Resolving the problem
To resolve the error, complete one of the following steps:

- Increase the FIFOOPENTIME setting by using the **Usage > Bulk load to DB2 on z/OS > Transfer > Transfer command** property.

- Set the **Transfer command** property to **quote site fifoopentime=***n*, where *n* is the number of seconds the FTP server waits for the LOAD utility to start.

You can find more information about the FIFOOPENTIME statement in the following topic: http://publib.boulder.ibm.com/infocenter/zos/v1r12/index.jsp?topic=%2Fcom.ibm.zos.r12.halz001%2Ffifoopentime.htm

Chapter 4. DB2 API stage

Use the IBM DB2 UDB API stage to access DB2 data from an IBM InfoSphere DataStage server job.

When you use IBM InfoSphere DataStage to access DB2 data from an IBM InfoSphere DataStage server job, you can choose from a collection of connectivity options. For most new jobs, use the DB2 Connector stage, which offers better functionality and performance than the DB2 API stage.

If you have jobs that use the DB2 API stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

Introduction

IBM DB2 API processes SQL statements in the native IBM DB2 environment. It also provides native importing of metadata definitions into the IBM InfoSphere DataStage Repository as well as live data browsing during job design.

The DB2 UDB API stage enables the InfoSphere DataStage to write data to and read data from an IBM DB2 database. The DB2 UDB API stage is passive and can have any number of input, output, and reference output links.

- **Input links.** Specify the data you are writing, which is a stream of rows to be loaded into an IBM DB2 database. You can specify the data on an input link by using an SQL statement generated by the InfoSphere DataStage or constructed by the user.
- **Output links.** Specify the data you are extracting, which is a stream of rows to be read from an IBM DB2 database. You can specify the data on an output link by using an SQL SELECT statement generated by the InfoSphere DataStage or constructed by the user.
- **Reference output links.** Represent rows that are read from an IBM DB2 database by using the key columns in a WHERE clause of the SELECT statement. These statements can be constructed by the InfoSphere DataStage or specified by the user. The key columns are determined by column definitions specified for the link.

In summary, the purpose of this plug-in is to eliminate the need for the ODBC stage in order to access IBM DB2 data by providing native capabilities for the following:

- Reading and writing data (DML)
- Creating and dropping tables (DDL)
- Importing table and column definitions (metadata)
- Browsing native data with the custom IBM DB2 property editor

Functionality of the DB2 UDB API stage

Some of the functionality of the DB2 UDB API stage are: automatically generates SQL statements to read or write IBM DB2 data and automatically drops and creates specified target tables.

The IBM DB2 plug-in has the following functionality:

- Connects to IBM DB2 on an AS/400® system by using the DRDA® protocol (TCP/IP).
- Uses stream input, stream output, and reference output links.
- Imports table and column definitions from the target IBM DB2 database and stores them in the Repository.
- Automatically generates SQL statements to read or write IBM DB2 data. (You can override this with user-defined SQL statements.)
- Automatically drops and creates specified target tables. (You can override this with user-defined SQL statements.)
- Uses file names to contain your SQL statements.
- Provides a custom user interface for editing the IBM DB2 plug-in properties.
- Uses stored procedures.
- Supports NLS (National Language Support).
- Allows data browsing through the custom property editor. You can use the custom GUI for the plug-in to view sample native table data residing on the target IBM DB2 database.
- Supports reject row handling.

The following functionality is not supported:

- Bulk loading of IBM DB2 tables from stream input. Although vast amounts of data be read into an IBM DB2 database by using this plug-in, the stream input links are not designed for performance-critical loading. You should use the DB2 UDB Load stage for this purpose.
- Replacing the ODBC stage. The IBM DB2 API stage does not replace the ODBC stage. The ODBC stage will continue to exist for access to data for which the IBM InfoSphere DataStage does not provide a native interface. Users who created jobs by using the ODBC stage to access an IBM DB2 database can continue to run these jobs.
- The large object family of IBM DB2 data types (BLOB and DBCLOB).

Installing the Stage

The installation of IBM DB2 API stage requires the DRDA protocol (TCP/IP) if the IBM DB2 data resides on an AS/400 system.

You can temporarily use the grid-style editor by right-clicking the plug-in icon and choosing **Grid Style** from the shortcut menu. Use this editor if you want to use job parameters for property values. (You cannot use the GUI to enter job parameters in boxes that require numeric values.)

Setting environment variables for IBM DB2 Database

You must set certain environment variables in order for the stage to work correctly on a UNIX platform.

About this task

Set the following environment variables on the server machine.

- DB2INSTANCE
- INSTHOME
- LD_LIBRARY_PATH

The actual name of the environment variable LD_LIBRARY_PATH differs depending on the platform.

- If the platform is AIX, use LIBPATH.
- If the platform is HP_UX, use SHLIB_PATH.
- If the platform is LINUX or Solaris, use LD_LIBRARY_PATH.

The IBM DB2 Database Connection

When you use the stage GUI to edit a DB2 UDB API stage, which is an easier method than using grids, the stage editor dialog box opens. This dialog box has **Stage**, **Input**, and **Output** pages depending on whether there are inputs to and outputs from the stage.

- **Stage.** This page displays the name of the stage you are editing. The **General** tab defines the IBM DB2 database server name, login information, and transaction isolation level information for concurrency control in jobs. You can describe the purpose of the stage in the **Description** field. The properties on this page define the connection to the data source. For details, see "Connecting to an IBM DB2 Data Source."

  The **NLS** tab defines a character set map to be used with the stage. This tab appears only if you have installed NLS for the InfoSphere DataStage. For details, see "Defining Character Set Mapping" .
- **Input.** This page is displayed only if you have an input link to this stage. It specifies the SQL table to use and the associated column definitions for each data input link. It also specifies how data is written and contains the SQL statement or call syntax used to write data to a table. It also specifies how to create the target table if desired and how to drop it if necessary.
- **Output.** This page is displayed only if you have an output or reference output link to this stage. It specifies the SQL tables to use and the associated column definitions for each data output link. It contains the SQL SELECT statement or call syntax used to read data from one or more tables or views.

Defining the IBM DB2 Database Connection

To define the IBM DB2 database connection, you need to define the data on the input and output links of the data source.

About this task

Perform the following steps to define a DB2 UDB API stage from the stage editor dialog box.

Procedure

1. Connect to a data source (see "Connecting to an IBM DB2 Data Source").
2. Optional: Define a character set map (see Defining Character Set Mapping).
3. Define the data on the input links (see Defining IBM DB2 Input Data).
4. Define the data on the output links (see Defining IBM DB2 Output Data).
5. Click **OK** to close this dialog box. Changes are saved when you save the job design.

Connecting to an IBM DB2 Data Source

The IBM DB2 connection parameters are set on the **General** tab of the Stage page.

About this task

Procedure

1. Enter the name of the server to access in the **Server name** field. Use the Client Configuration Assistant on Windows (Windows 2000 or Windows Server 2003) or the command line processor in UNIX to configure the IBM DB2 database server on the IBM DB2 database client, which is the InfoSphere DataStage server. This is a required field with no default.
2. Optionally, enter the name to use to connect to the database in the **User ID** field. (Without a user name in this field, the database uses the user name of the process running the InfoSphere DataStage job to connect to the server.)

   This user must have sufficient privileges to access the specified database and source and target tables.
3. Enter the optional password that is associated with the specified user name to use in the **Password** field. For security, it displays asterisks instead of the value you enter. There is no default. If **User ID** is omitted, this field is ignored.
4. Choose an appropriate transaction isolation level to use from the **Transaction Isolation** list (see "Transaction Isolation Levels"). These levels provide the necessary concurrency control between transactions in the job and other transactions. You cannot edit this field, which is required.
5. Optionally, describe the purpose of the IBM DB2 API stage in the **Description** field

Transaction Isolation Levels

The transaction isolation levels of IBM DB2 are: uncommitted read, cursor stability, read stability, and repeatable read.

Uncommitted Read. Takes exclusive locks on modified data. This level is equivalent to read uncommitted. These locks are held until a commit or rollback is executed. However, other transactions can still read but not modify the uncommitted changes. No other locks are taken.

Cursor Stability. Takes exclusive locks on modified data and sharable locks on all other data. This is the default. This level is equivalent to read committed. Exclusive locks are held until a commit or rollback is executed. Uncommitted changes are not readable by other transactions. Shared locks are released immediately after the data has been processed, allowing other transactions to modify it.

Read Stability. Identical to repeatable read except that phantom rows might be seen.

Repeatable Read. Equivalent to serializable. This level takes exclusive locks on modified data and sharable locks on all other data. All locks are held until a commit or rollback is executed, preventing other transactions from modifying any data that has been referenced during the transaction.

The IBM DB2 terminology for transaction isolation levels differs from ANSI terminology. Therefore, the options differ from those found in other stages such as the Informix® CLI stage.

Defining Character Set Mapping

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

About this task

The default character set map is defined for the project or the job. You can change the map by selecting a map name from the list.

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

Show all maps lists all the maps that are shipped with the IBM InfoSphere DataStage. **Loaded maps only** lists only the maps that are currently loaded.

Defining IBM DB2 Input Data

When you write data to a table in an IBM DB2 database, the DB2 UDB API stage has an input link.

Define the properties of this link and the column definitions of the data on the Input page in the stage editor dialog box.

General Tab

Use this tab to indicate how the SQL statements are created from an **Input** link on the DB2 UDB API stage.

This tab is displayed by default. It contains the following fields:

- **Query Type.** Determines how the SQL statements are created. Options are
  - **Use SQL Builder tool.** Causes the **SQL Builder** button and the **Update action** property to appear. This is the default value for new jobs.
  - **Generate Update action from Options and Columns tabs.** Causes the **Update action** property to appear. Uses values from the **Options** and **Columns** tabs and from **Update action** to generate the SQL.
  - **Enter custom SQL statement.** Writes the data by using a user-defined SQL statement, which overrides the default SQL statement generated by the stage. If you choose this option, you enter the SQL statement on the SQL tab.
- **SQL Builder.** Causes SQL Builder to open.
- **Update action.** Specifies which stage-generated SQL statements are used to update the target table. Some update actions require key columns to update or delete rows. The default is **Insert rows without clearing**. Choose one of the following options:
  - **Insert rows without clearing.** Inserts the new rows in the table. When you click **SQL Button**, the Insert page opens.
  - **Clear the table, then insert rows.** Deletes the contents of the table before inserting the new rows. When you click **SQL Button**, the Insert page opens.
  - **Delete existing rows only.** Deletes existing rows in the target file that have identical keys in the input rows. When you click **SQL Button**, the Delete page opens.

- **Replace existing rows completely.** Deletes the existing rows, then adds the new rows to the table. When you click **SQL Button**, the **Delete** page opens. However, you must also complete an Insert page to accomplish the replace.
- **Update existing rows only.** Updates the existing data rows. Any rows in the data that do not exist in the table are ignored. When you click **SQL Button**, the Update page opens.
- **Update existing or insert new rows.** Updates the existing data rows before inserting new rows. Performance depends on the contents of the target table and the rows being processed in the job. If most rows exist in the target table, it is faster to update first. When you click **SQL Button**, the Update page opens. However, you must also complete an Insert page to accomplish the replace.
- **Insert new or update existing rows.** Inserts the new rows before updating existing rows. Performance depends on the contents of the target table and the rows being processed in the job. If most rows do not exist in the target table, it is faster to insert first. When you click **SQL Button**, the Insert page opens. However you must also complete an Update page to accomplish the update.

  Note: If using **Update existing or insert new rows** or **Insert new or update existing rows** for the update action, **Array size**, located on the Options tab, must be 1. (Otherwise, a warning is logged and the stage automatically sets it to 1.)

Options tab

Use the **Options** tab to create or drop tables and to specify miscellaneous link options.

- **Table name.** It is the name of the target table to update. You must specify the target table. There is no default. You can also click the **...** button at the right of **Table name** to browse the Repository to select the table.
- **Create table action.** Choose one of the following options to create the target table in the specified database:
  - **Do not create target table.** Specifies that the target table is not created, and the **Drop table action** field and the **Table Properties** button (at the right of the field) are disabled. If the target table does not exist when you run a job, the job aborts.
  - **Generate DDL.** Specifies that the stage generates the CREATE TABLE statement by using the information obtained from the **Table name** field, the column definitions grid, and the advanced table properties (see the description for the **Table Properties** button later in this section). If the target table already exists, the job aborts.
  - **User-defined DDL.** Specifies that you enter the appropriate CREATE TABLE statement on the **SQL** tab. You can customize the stage-generated DDL that the stage provides as a template. If the target table already exists, the job aborts.
- **Drop table action.** Lets you control the dropping of the target table before it is created by the stage. If you choose not to create the target table, this field is disabled. Choose one of the following options:
  - **Do not drop target.** Specifies that the target table is not dropped.
  - **Generate DDL.** Specifies that the stage generates DDL based on the value of the **Table name** field. If the target table does not exist, a warning is logged. The job does not abort.

  - **User-defined DDL.** Specifies that you should define the DDL to drop the target table. You can customize the stage-generated DDL that the stage provides as a template. If the target table does not exist, a warning is logged. The job does not abort.
- **Table Properties button.** Click the button at the right of the **Drop table action** list box to display the Create Table Properties dialog box. (This button is enabled when you select **Generate DDL** or **User-defined DDL** from the **Create table action** list box.) You can then specify the following advanced table properties from this dialog box.
  - **Tablespace.** Specifies an existing tablespace name. The new table is created in this tablespace. If you omit the name, the table is created in the default tablespace as defined by the database.
  - **Partitioning Key.** Specifies the columns to use for partitioning the data for a table in a multi partitioned node group. If you omit this field and the table resides in a multi partitioned node group, the table is partitioned by using the default partitioning rules as defined by the database**.**
- **Array size.** The input parameter array size. The default is 50 rows that are cached before being written to the database. The array size value should be an integer greater than or equal to 1. If a table that is being updated is also being used for reference lookups, **Array size** must be 1 so the updates can be referenced.
- **Transaction size.** The number of rows that the stage processes before committing a transaction to the database. The transaction size should always be a multiple of the array size. The default is 100. The transaction size should be an integer greater than or equal to 0. A value of 0 means that the transaction will not be committed until all rows have been processed.

Columns Tab

This tab contains the column definitions for the data written to the table or file.

The **Columns** tab behaves the same way as the **Columns** tab in the ODBC stage.

SQL Tab

Use this tab to display the stage-generated SQL statement and the SQL statement that you can enter.

This SQL tab contains the following tabs:

- **Query.** This tab is displayed by default. It is similar to the **General** tab, but it contains the SQL statements that are used to write data to Oracle. It is based on the current values of the stage and link properties. You cannot edit these statements unless **Query type** is set to **Enter custom SQL statement** or **Load SQL from a file at run time**.
- **Before.** This tab contains the SQL statements executed before the stage processes any job data rows. The elements on this tab correspond to the Before SQL and Continue if Before SQL fails grid properties. The **Before** and **After** tabs look alike. The **Continue if Before SQL fails** property is represented by a check box and the SQL statement is entered in an edit box that you can resize.
- **After.** This tab contains the SQL statements executed after the stage processes job data rows. The elements on this tab correspond to the After SQL and Continue if After SQL fails grid properties. The **Before** and **After** tabs look alike. The **Continue if After SQL fails** property is represented by a check box and the SQL statement is entered in a an edit box that you can resize.

- **Generated DDL**. Select **Generate DDL** or **User-defined DDL** from the **Create table action** field on the **General** tab to enable this tab. The **CREATE statement** field displays the non-editable CREATE TABLE statement that is generated from the column metadata definitions and the information provided on the Create Table Properties dialog box. If you select an option other than **Do not drop target table** from the **Drop table action** list, the **DROP statement** field displays the generated DROP TABLE statement for dropping the target table.
- **User-defined DDL.** Select **User-defined DDL** from the **Create table action** or **Drop table action** field on the **General** tab to enable this tab. The generated DDL statement is displayed as a starting point from which you can define a CREATE TABLE and a DROP TABLE statement.

  The **DROP statement** field is disabled if **User-defined DDL** is not selected from the **Drop table action** field. If **Do not drop target** is selected, the **DROP statement** field is empty in the **Generated DDL** and **User-defined DDL** tabs.

  Note: Once you modify the user-defined DDL statement from the original generated DDL statement, changes made to other table-related properties do not affect the user-defined DDL statement. If, for example, you add a new column in the column grid after modifying the user-defined DDL statement, the new column appears in the generated DDL statement but does not appear in the user-defined DDL statement. You must ensure that the user-defined SQL results in the creation or dropping of the correct target table.

Defining IBM DB2 Output Data

When you read data from an IBM DB2 data source, the DB2 UDB API stage has an output link.

The properties of this link and the column definitions of the data are defined on the Output page in the stage editor dialog box.

General Tab

This tab provides the type of query and, where appropriate, a button to open an associated dialog box. It is displayed by default.

The **General** tab contains the following field:

- **Query type.** Displays the following options.
  - **Use SQL Builder Tool.** Specifies that the SQL statement is built using the SQL Builder graphical interface. When this option is selected, the **SQL Builder** button appears. If you click **SQL Builder**, the SQL Builder opens. See *IBM InfoSphere DataStage and QualityStage Designer Client Guide* for a complete description of the SQL Builder. This is the default setting.
  - **Generate SELECT clause from column list; enter other clauses.** Specifies that the InfoSphere DataStage generates the SELECT clause based on the columns you select on the **Columns** tab. When this option is selected, the **SQL Clauses** button appears. If you click **SQL Clauses**, the SQL Clauses dialog box appears. Use this dialog box to refine the SQL statement.
  - **Enter custom SQL statement.** Specifies that a custom SQL statement is built by using the **SQL** tab. See "SQL Tab" .
  - **Load SQL from a file at run time.** Specifies that the data is extracted by using the SQL query in the path name of the designated file that exists on the server. Enter the path name for this file instead of the text for the query. With this choice, you can edit the SQL statements.

- **Description.** Lets you enter an optional description of the output link.

Options Tab

This tab provides the number of prefetch rows.

- **Prefetch rows.** The number of rows that the database returns when the IBM InfoSphere DataStage fetches data from the source tables. Specifying a value greater than 1 improves performance (memory usage increases to accommodate buffering multiple rows). The value should be an integer greater than or equal to 1.

Columns Tab

This tab contains the column definitions for the data being output on the chosen link.

Enter the appropriate table name in the **Description** field on output links to qualify column references. Do this if any ambiguity exists as to which table the indicated columns belong.

The column definitions for reference links require a key field. Key fields join reference inputs to a Transformer stage. The key reads the data by using a WHERE clause in the SQL SELECT statement.

SQL Tab

This tab displays the stage-generated or user-defined SQL statements or stored procedure call syntax used to read data from a table.

It contains the **Query**, **Before**, and **After** tabs.

- **Query.** This tab is read-only if you select **Use SQL Builder tool** or **Generate SELECT clause from column list; enter other clauses** for **Query Type**. If **Query Type** is **Enter Custom SQL statement**, this tab contains the SQL statements executed to read data from a table. The GUI displays the stage-generated SQL statement on this tab as a starting point. However, you can enter any valid, appropriate SQL statement. If **Query Type** is **Load SQL from a file at run time**, enter the path name of the file.
- **Before.** This tab contains the SQL statements executed before the stage processes any job data rows.
- **After.** This tab contains the SQL statements executed after the stage processes all job data rows.

Data Type Support

You can perform mapping from the IBM InfoSphere DataStage SQL data types to IBM DB2 SQL data types and the mapping from IBM DB2 SQL data types to the InfoSphere DataStage SQL types.

Mapping Data Types from IBM InfoSphere DataStage SQL to IBM DB2 SQL

When the Create Table property is set to Yes for input links, the target table is created using the column definitions for the input link and the specific input link properties defining the target table's properties. In some cases, there is no exact translation between a IBM DB2 data type and an InfoSphere DataStage data type, for example, GRAPHIC.

The following table shows the IBM DB2 data types that are generated from the corresponding InfoSphere DataStage types:

Table 11. InfoSphere DataStage data types and corresponding IBM DB2 data types

InfoSphere DataStage SQL Data Type	IBM DB2 SQL Data Type
SQL_BIGINT	BIGINT
SQL_BINARY	CHAR FOR BIT DATA
SQL_BIT	Unsupported
SQL_CHAR	CHAR
SQL_DATE	DATE
SQL_DECIMAL	DECIMAL
SQL_DOUBLE	DOUBLE PRECISION
SQL_FLOAT	FLOAT
SQL_INTEGER	INTEGER
SQL_LONGVARBINARY	LONG VARCHAR FOR BIT DATA
SQL_LONGVARCHAR	LONG VARCHAR
SQL_LONGVARCHAR	CLOB (see note below)
SQL_NUMERIC	DECIMAL
SQL_REAL	REAL
SQL_SMALLINT	SMALLINT
SQL_TIME	TIME
SQL_TIMESTAMP	TIMESTAMP
SQL_TINYINT	SMALLINT
SQL_VARBINARY	VARCHAR FOR BIT DATA
SQL_VARCHAR	VARCHAR

Note: The DB2 UDB API stage supports the CLOB data type by mapping the LONGVARCHAR data type with a precision greater than 32 K to IBM DB2's CLOB data type. To work with a CLOB column definition, choose the InfoSphere DataStage's LONGVARCHAR as the column's data type and provide a Length of more than 32 K in the **Columns** tab. If the Length is less than or equal to 32 K, the InfoSphere DataStage's LONGVARCHAR maps to LONGVARCHAR.

Mapping Data Types from IBM DB2 SQL to IBM InfoSphere DataStage SQL

When the DB2 UDB API stage imports metadata definitions from a database, it must perform a mapping of the database SQL data types to the SQL data types supported by the InfoSphere DataStage

The following table describes the mapping between the IBM DB2 SQL data types and the InfoSphere DataStage SQL data types:

Table 12. IBM DB2 data types and the corresponding InfoSphere DataStage data types

IBM DB2 SQL Data Type	InfoSphere DataStage SQL Data Type
BIGINT	SQL_BIGINT
CHAR	SQL_CHAR

Table 12. IBM DB2 data types and the corresponding InfoSphere DataStage data types (continued)

IBM DB2 SQL Data Type	InfoSphere DataStage SQL Data Type
CHAR FOR BIT DATA	SQL_BINARY
DATE	SQL_DATE
DECIMAL	SQL_DECIMAL
DOUBLE PRECISION	SQL_DOUBLE
FLOAT	SQL_FLOAT
GRAPHIC	SQL_CHAR
INTEGER	SQL_INTEGER
LONG VARCHAR	SQL_LONGVARCHAR
LONG VARCHAR FOR BIT DATA	SQL_LONGVARBINARY
LONG VARGRAPHIC	SQL_LONGVARCHAR
NUMERIC	SQL_NUMERIC
REAL	SQL_REAL
SMALLINT	SQL_SMALLINT
TIME	SQL_TIME
TIMESTAMP	SQL_TIMESTAMP
VARCHAR	SQL_VARCHAR
VARCHAR FOR BIT DATA	SQL_VARBINARY
BLOB and LOCATOR	Unsupported
CLOB and LOCATOR	SQL_LONGVARCHAR (see note below)
DBCLOB and LOCATOR	Unsupported

Note: The DB2 UDB API stage supports the CLOB data type by mapping the LONGVARCHAR data type with a precision greater than 32 K to IBM DB2's CLOB data type. To work with a CLOB column definition, choose the InfoSphere DataStage's LONGVARCHAR as the column's data type and provide a Length of more than 32 K in the **Columns** tab. If the Length is less than or equal to 32 K, the InfoSphere DataStage's LONGVARCHAR maps to LONGVARCHAR.

Handling $ and # Characters

IBM InfoSphere DataStage has been modified to enable it to handle databases which use the reserved characters # and $ in column names.

About this task

InfoSphere DataStage converts these characters into an internal format, then converts them back as necessary.

To take advantage of this facility, do the following:

- Avoid using the strings __035__ and __036__ in your column names (these are used as the internal representations of # and $ respectively).
- When using this feature in your job, you should import meta data using the Plug-in Meta Data Import tool, and avoid hand-editing (this minimizes the risk of mistakes or confusion).

Once the table definition is loaded, the internal column names are displayed rather than the original IBM DB2 database names both in table definitions and in the Data Browser. They are also used in derivations and expressions. The original names are used in generated SQL statements, however, and you should use them if entering SQL in the job yourself.

When using an DB2 UDB API stage in a server job, you should use the external names when entering SQL statements that contain IBM DB2 database columns. The columns within the stage are represented by question marks (parameter markers) and bound to the columns by order, so you do not need to worry about entering names for them. This applies to:

- Query
- Update
- Insert
- Key
- Select
- Where clause

For example, for an update you might enter:

```
UPDATE tablename SET ##B$ = ? WHERE  $A# = ?
```

Particularly note the key in this statement ($A#) is specified by using the external name.

Troubleshooting

If your source data is defined correctly, rows are properly inserted in a target table. However, under certain conditions rows might not be inserted in a target table.

IBM DB2 rejects the remainder of the row batch following a 'bad' row when the following three conditions occur:

- The Array Size property exceeds 1.
- The defined string lengths of source data exceed the defined length of its target column.
- The source data contains a row with a character string that exceeds the length of the target column.

Erroneous error messages concerning those remaining rows also result.

Example

Suppose the target table defines a column as CHAR(5), the IBM InfoSphere DataStage metadata for this column is defined as CHAR(10), and the source data contains the following rows:

'ABC'

'ABCD'

'ABCDEFG' (Longer than 5 characters)

'AB'

'ABCD'

The last three rows are not inserted into the target table when the Array Size property is set to 5. IBM DB2 reports that all three rows contained values that were too large (IBM DB2 error SQL0302N).

Additionally, using BIGINT for source data that contains out-of-range values causes similar behavior.

Solutions

Define the InfoSphere DataStage metadata correctly to match the IBM DB2 target, and ensure that any BIGINT source data is within BIGINT range. Otherwise, it might be safer to run the job with Array Size set to 1. However, this can impact performance.

Another solution is to use a Transformer stage to scrub the data before sending it to the DB2 UDB API stage. This method also impacts performance.

Note: This behavior does not occur for rows rejected by the database for reasons such as constraint violations or non-null violations. The remaining rows in a batch are not rejected.

Chapter 5. DB2 UDB Enterprise stage

Use the IBM DB2 UDB Enterprise stage to access DB2 data from an IBM InfoSphere DataStage parallel job.

When you use IBM InfoSphere DataStage to access DB2 data from an IBM InfoSphere DataStage parallel job, you can choose from a collection of connectivity options. For most new jobs, use the DB2 Connector stage, which offers better functionality and performance than the DB2 UDB Enterprise stage.

If you have jobs that use the DB2 UDB Enterprise stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

Overview

The DB2/UDB enterprise stage is a database stage. By using DB2/UDB enterprise stage, you can read data from and write data to an IBM DB2 database. You can use the stage in conjunction with a lookup stage to access a lookup table hosted by an IBM DB2 database. See the *IBM InfoSphere DataStage and QualityStage Parallel Job Developer's Guide*.

IBM DB2 databases distribute data in multiple partitions. DB2/UDB enterprise stage can match the partitioning when reading data from or writing data to an IBM DB2 database.

Depending upon the properties that you set for the DB2/UDB enterprise stage, the stage can have:

- One input link for the load and write methods and
- One output link for the write method or a reference output link. The lookup stage uses the reference output link when referring to a IBM DB2 lookup table. Alternatively DB2/UDB enterprise stage can have a single output reject link (in conjunction with an input link).

By using DB2/UDB enterprise stage, you can perform the following operations:

- Writing data to an IBM DB2 table (by using INSERT).
- Updating an IBM DB2 table (by using INSERT and/or UPDATE as appropriate), by using DB2 command line interface (CLI) to enhance performance.
- Loading data to an IBM DB2 table (by using DB2 fast loader). (Note loading is not supported by mainframe DB2 databases.)
- Reading data from an IBM DB2 table.
- Deleting rows from an IBM DB2 table.
- Performing a lookup operation directly on an IBM DB2 table.
- Loading an IBM DB2 table into memory and then perform a lookup on that IBM DB2 table.

When using a DB2/UDB enterprise stage as a source for lookup data, there are special considerations about column naming. If you have columns of the same name in both the source and lookup data sets, the source data set column will go to the output data. If you want this column to be replaced by the column from the lookup data source, you need to drop the source data column before you perform

the lookup (you could, for example, use a Modify stage to do this). See the topic on Merge stages in the *IBM InfoSphere DataStage and QualityStage Parallel Job Advanced Developer's Guide* for more details about performing lookup operations.

To edit a DB2/UDB enterprise stage, you use the stage editor. To learn about the stage editor in detail, the *Parallel Job Developer's Guide*.

Accessing IBM DB2 Databases

Before you begin

Before using DB2/UDB enterprise stage for the first time, you should complete the configuration procedures. Refer to the InfoSphere DataStage PDF documentation.

About this task

To use the DB2/UDB enterprise stage, you must have valid accounts and appropriate privileges on the databases to which the stage connects. If you are using IBM DB2 8.1 Enterprise Server Edition (ESE), it is recommended that you install DPF in order to leverage the InfoSphere DataStage's parallel capabilities.

Listed below are the required IBM DB2 privileges:

- SELECT privilege on any tables from which to read data.
- INSERT privilege on any existing tables to be updated.
- TABLE CREATE privilege to create any new tables.
- INSERT and TABLE CREATE privileges on any existing tables to be replaced.
- DBADM privilege on any database written by using the LOAD method.

You can grant this privilege in several ways in IBM DB2. One is to start IBM DB2, connect to a database, and grant DBADM privilege to a user, as shown below:

```
db2> CONNECT TO db_namedb2> GRANT DBADM ON DATABASE TO USER user_name
```

where *db_name* is the name of the IBM DB2 database and *user_name* is your InfoSphere DataStage login user name. If you specify the **message file** property in conjunction with LOAD method, the database instance must have read or write privilege on that file. The location of the log file for the LOAD operation messages is exactly as defined in the APT_CONFIG_FILE.

Your PATH should include $*DB2_HOME/bin*, for example, */opt/IBMdb2/V8.1/bin*. The LIBPATH should include $DB2_HOME/lib before any other lib statements, for example, */opt/IBMdb2/V8.1/lib*.

The following IBM DB2 environment variables set the runtime characteristics of your system:

- DB2INSTANCE specifies the user name of the owner of the IBM DB2 instance. IBM DB2 uses DB2INSTANCE to determine the location of *db2nodes.cfg*. For example, if you set DB2INSTANCE to "Mary", the location of *db2nodes.cfg* is *~Mary/sqllib/db2nodes.cfg*.
- DB2DBDFT specifies the name of the IBM DB2 database that you want to access from your DB2/UDB enterprise stage.

There are two other methods of specifying the IBM DB2 database:

1. The **override database** property of the DB2/UDB enterprise stage Input or Output link.

2. The APT_DBNAME environment variable (this takes precedence over DB2DBDFT).

You should normally use the input property **Row Commit Interval** to specify the number of records to insert into a table between commits (see the **Row Commit Interval** section under **Options** category). Previously the environment variable APT_RDBMS_COMMIT_ROWS was used for this, and this is still available for backwards compatibility. You can set this environment variable to any value between 1 and (2^{31} - 1) to specify the number of records. The default value is 2000. If you set APT_RDBMS_COMMIT_ROWS to 0, a negative number, or an invalid value, a warning is issued and each partition commits only once after the last insertion.

If you set APT_RDBMS_COMMIT_ROWS to a small value, you force IBM DB2 to perform frequent commits. Therefore, if your program terminates unexpectedly, your data set can still contain partial results that you can use. However, the high frequency of commits might affect performance of DB2/UDB enterprise stage. If you set a large value for APT_RDBMS_COMMIT_ROWS, DB2 must log a correspondingly large amount of rollback information. This, too, might slow your application.

Remote connection

About this task

You can also connect from a DB2/UDB enterprise stage to a remote IBM DB2 Server. The connection is made via an IBM DB2 client.

In order to remotely connect from an IBM DB2 client to an IBM DB2 server, the IBM DB2 client should be located on the same machine as the InfoSphere DataStage server. Both IBM DB2 client and IBM DB2 server need to be configured for remote connection communication (see your IBM DB2 Database Administrator).

The InfoSphere DataStage configuration file needs to contain the node on which the InfoSphere DataStage and the IBM DB2 client are installed and the nodes of the remote computer where the IBM DB2 server is installed. See the topic about the parallel engine configuration file in the *IBM InfoSphere DataStage and QualityStage Parallel Job Developer's Guide*.

Procedure

On the DB2/UDB enterprise stage in your parallel job, you need to set the following properties:

- **Client Instance Name**. Set this to the IBM DB2 client instance name. If you set this property, the InfoSphere DataStage assumes you require remote connection.
- **Server**. Optionally set this to the instance name of the IBM DB2 server. Otherwise use the IBM DB2 environment variable, DB2INSTANCE, to identify the instance name of the IBM DB2 server.
- **Client Alias DB Name**. Set this to the IBM DB2 client's alias database name for the remote IBM DB2 server database. This is required only if the client's alias is different from the actual name of the remote server database.
- **Database**. Optionally set this to the remote server database name. Otherwise use the environment variables APT_DBNAME or APT_DB2DBDFT to identify the database.

- **User**. Enter the user name for connecting to IBM DB2. This is required for a remote connection.
- **Password**. Enter the password for connecting to IBM DB2. This is required for a remote connection.

You can use the remote connection facilities available in the InfoSphere DataStage to connect to a different IBM DB2 server within the same job. You could, for example, read from an IBM DB2 database on one server, use this data to access a lookup table on another IBM DB2 server, and then write any rejected rows to a third IBM DB2 server. Each database would be accessed by a different stage in the job with the Client Instance Name and Server properties set appropriately.

Handling special characters # and $

About this task

The characters # and $ are reserved in the IBM InfoSphere DataStage. Special steps are needed to handle IBM DB2 databases which use the characters # and $ in column names. InfoSphere DataStage converts these characters into an internal format, and then converts them back as necessary.

To take advantage of this facility, you need to do the following:

- In the IBM InfoSphere DataStage and QualityStage Administrator client, open the Environment Variables dialog for the project in question, and set the environment variable DS_ENABLE_RESERVED_CHAR_CONVERT to true (this can be found in the General\Customize branch).
- Avoid using the strings __035__ and __036__ in your IBM DB2 column names. These strings are used as the internal representations of # and $ respectively.

When using this feature in your job, you should import metadata by using the Plug-in Meta Data Import tool, and avoid manual editing. This minimizes the risk of mistakes or confusion.

Once the table definition is loaded, the internal column names are displayed rather than the original IBM DB2 names both in table definitions and in the Data Browser. The column names are also used in derivations and expressions. The original names are used in generated SQL statements, however, and you should use them if entering SQL in the job yourself.

Generally, in the DB2/UDB enterprise stage, you enter external names everywhere except when referring to stage column names, where you use names in the form ORCHESTRATE.*internal_name*.

When using the DB2/UDB enterprise stage as a target, you should enter external names as follows:

- For Write and Load options, use external names for select list properties.
- For Upsert option, for update and insert, use external names when referring to IBM DB2 table column names, and internal names when referring to the stage column names. For example:

```
INSERT INTO tablename ($A#, ##B$) VALUES
(ORCHESTRATE.__036__A__035__, ORCHESTRATE.__035__035__B__036__)

UPDATE tablename SET ##B$ = ORCHESTRATE.__035__035__B__036__ WHERE ($A# =
ORCHESTRATE.__036__A__035__)
```

When using the DB2/UDB enterprise stage as a source, you should enter external names as follows:

- For Read using the user-defined SQL method, use external names for IBM DB2 columns for SELECT: For example:

```
SELECT #M$, #D$ FROM tablename WHERE (#M$ > 5)
```

- For Read using Table method, use external names in select list and where properties.

When using the DB2/UDB enterprise stage in parallel jobs as a lookup, you should enter external or internal names as follows:

- For Lookups using the user-defined SQL method, use external names for IBM DB2 columns for SELECT, and for IBM DB2 columns in any WHERE clause you might add. Use internal names when referring to the stage column names in the WHERE clause. For example:

```
SELECT #M$, #D$ FROM tablename
WHERE (#B$ = ORCHESTRATE.__035__ B __036__)
```

- For Lookups using the Table method, use external names in select list and where properties.
- Use internal names for the key option on the Input page **Properties** tab of the Lookup stage to which the DB2/UDB enterprise stage is attached.

Using the Pad Character property

About this task

Use the **Pad Character** property when using upsert or performing a lookup to pad string and ustring fields that are less than the length of the DB2 CHAR column. Use this property for string and ustring fields that are inserted in IBM DB2 or are used in the WHERE clause of an UPDATE, DELETE, or SELECT statement when all three of these conditions are met:

1. The UPDATE or SELECT statement contains string or ustring fields that map to CHAR columns in the WHERE clause.
2. The length of the string or ustring field is less than the length of the CHAR column.
3. The padding character for the CHAR columns is not the null terminator.

For example, if you add rows to a table by using an INSERT statement in SQL, IBM DB2 automatically pads CHAR fields with spaces. When you subsequently use the DB2/UDB enterprise stage to update or query the table, you must use the Pad Character property with the value of a space in order to produce the correct results.

When you insert rows and subsequently update or query them by using the DB2/UDB enterprise stage, you do not need to specify the **Pad Character** property. The stage automatically pads with null terminators, and the default pad character for the stage is the null terminator.

Data type conversion - writing data to an IBM DB2 database

When writing or loading, DB2/UDB enterprise stage automatically converts InfoSphere DataStage data types to IBM DB2 data types as shown in the following table:

Table 13. Data type conversion - writing data to IBM DB2 Databases

InfoSphere DataStage SQL Data Type	**Underlying Data Type**	**IBM DB2 Data Type**
InfoSphere DataStage SQL Data Type	Underlying Data Type	IBM DB2 Data Type
Date	date	DATE
Time	time	TIME
Timestamp	timestamp	TIMESTAMP
Decimal Numeric	decimal (*p, s*)	DECIMAL (*p, s*)
tinyInt	int8	SMALLINT
SmallInt	int16	SMALLINT
Integer	int32	INTEGER
Float Real	sfloat	FLOAT
Double	dfloat	FLOAT
Unknown Char	fixed-length string in the form string[*n*] and ustring[*n*]; length <= 254 bytes	CHAR(*n*) where *n* is the string length
LongVarChar VarChar	fixed-length string in the form string[*n*] and ustring[*n*]; length < 32672 for Varchar length < 32700 for LongVarchar	VARCHAR(*n*) where *n* is the string length
LongVarChar VarChar	variable-length string, in the form string[max=*n*] and ustring[max=*n*]; maximum length <= 4000 bytes	VARCHAR(*n*) where *n* is the maximum string length
LongVarChar VarChar	variable-length string in the form string and ustring	VARCHAR(32)*
LongVarChar VarChar	string and ustring, 4000 bytes < length	Not supported
NChar	ustring [*n*]	GRAPHIC
NVarChar	ustring [max = *n*]	VARGRAPHIC
LongNVarChar	ustring [max = *n*]	LONGVARGRAPHIC

Note: The default length of VARCHAR is 32 bytes. That is, 32 bytes are allocated for each variable-length string field in the input data set. If an input variable-length string field is longer than 32 bytes, the stage issues a warning.

Data type conversion - reading data from IBM DB2 Databases

When reading, the DB2/UDB enterprise stage automatically converts IBM DB2 data types to InfoSphere DataStage data types as shown in the following table:

Table 14. Data type conversion - reading data from IBM DB2 Databases

InfoSphere DataStage SQL Data Type	**Underlying Data Type**	**IBM DB2 Data Type**
InfoSphere DataStage SQL Data Type	Underlying Data Type	IBM DB2 Data Type
Time or Timestamp	time or timestamp with corresponding fractional precision for time If the DATETIME starts with a year component, the result is a timestamp field. If the DATETIME starts with an hour, the result is a time field.	DATETIME
Decimal Numeric	decimal (*p*, *s*) where p is the precision and s is the scale The maximum precision is 32, and a decimal with floating scale is converted to a dfloat	DECIMAL (*p*, *s*)
TinyInt	int8	SMALLINT
SmallInt	int16	SMALLINT
Integer	int32	INTEGER
Double	dfloat	FLOAT
Float Real	sfloat	SMALLFLOAT
Float Real	sfloat	REAL
Double	dfloat	DOUBLE-PRECISION
Decimal	decimal	MONEY
Unknown Char LongVarChar VarChar NChar NVarChar LongNVarChar	string[*n*] or ustring[*n*]	GRAPHIC(*n*)
Unknown Char LongVarChar VarChar NChar NVarChar LongNVarChar	string[max = *n*] or ustring[max = *n*]	VARGRAPHIC(*n*)
Unknown Char LongVarChar VarChar NChar NVarChar LongNVarChar	string[max = *n*] or ustring[max = *n*]	VARCHAR(*n*)
NChar	ustring [*n*]	GRAPHIC

Table 14. Data type conversion - reading data from IBM DB2 Databases (continued)

InfoSphere DataStage SQL Data Type	Underlying Data Type	IBM DB2 Data Type
NVarChar	ustring [max = *n*]	VARGRAPHIC
LongNVarChar	ustring [max = *n*]	LONGVARGRAPHIC

Examples

These examples show how to perform a lookup operation and how to update data in an IBM DB2 table.

Looking up an IBM DB2 table

This example shows what happens when data is looked up in an IBM DB2 table. The stage in this case will look up the interest rate for each customer based on the account type. The table below shows the data that arrives on the primary link:

Table 15. Example of a lookup operation - Table 1

Customer	accountNo	accountType	balance
Latimer	7125678	plat	7890.76
Ridley	7238892	flexi	234.88
Cranmer	7611236	gold	1288.00
Hooper	7176672	flexi	3456.99
Moore	7146789	gold	424.76

The table below shows the data in the IBM DB2 lookup table:

Table 16. Example of a lookup operation - Table 2

accountType	InterestRate
bronze	1.25
silver	1.50
gold	1.75
plat	2.00
flexi	1.88
fixterm	3.00

Here is what the lookup stage will output:

Table 17. Example of a lookup operation - Table 3

Customer	accountNo	accountType	balance	InterestRate
Latimer	7125678	plat	7890.76	2.00
Ridley	7238892	flexi	234.88	1.88
Cranmer	7611236	gold	1288.00	1.75
Hooper	7176672	flexi	3456.99	1.88
Moore	7146789	gold	424.76	1.75

The job looks like the jobs illustrated under the **Overview** section. When you edit a DB2/UDB enterprise stage, the Data_set stage provides the primary input, DB2_lookup_table provides the lookup data, Lookup_1 performs the lookup and outputs the resulting data to Data_Set_3. In the IBM DB2 database stage, you can specify to look up the data directly in the IBM DB2 database, and the name of the table. In the Lookup stage, you can specify the column that will be used as the key for the lookup.

Updating an IBM DB2 table

This example shows an IBM DB2 table being updated with three new columns. The database records the horse health records of a large stud. Details of the worming records are being added to the main table and populated with the most recent data, using the existing column "name" as a key. The metadata for the new columns is as follows:

Table 18. Metadata for example

Column name	Type
name	char
wormer_type	char
dose_interval	char
dose_level	char

You are going to specify upsert as the write method and choose User-defined Update and Insert as the upsert mode, this is so that we do not include the existing name column in the INSERT statement. The properties (showing the INSERT statement) are shown below. The INSERT statement is as generated by InfoSphere DataStage, except the name column is removed. The UPDATE statement is as automatically generated by InfoSphere DataStage:

```
UPDATE
horse_health
SET
wormer_type=ORCHESTRATE.wormer_type,
dose_interval=ORCHESTRATE.dose_interval,
WHERE
(name=ORCHESTRATE.name)
```

Must Do's

IBM InfoSphere DataStage has many defaults which means that it can be very easy to include DB2/UDB enterprise stages in a job. This section specifies the minimum steps to take to get a DB2/UDB enterprise stage functioning. InfoSphere DataStage provides a versatile user interface, and there are many shortcuts to achieving a particular end. This section describes the basic method. You will learn where the shortcuts are when you get familiar with the product.

Writing data to an IBM DB2 database

Procedure

1. In the **Input Link Properties** tab:
   a. Choose **Write** as the **Write Method**.
   b. Specify the name of the table you are writing.
   c. If you are not using environment variables to specify the server and database (as described in the **Accessing IBM DB2 Databases** section), set

Use Database Environment Variable and Use Server Environment Variable to False, and supply values for the Database and Server properties.

2. By default the stage partitions data in the same way as data is partitioned within an IBM DB2 table (see the **Accessing IBM DB2 Databases** section). You can change the partitioning method or specify a different database on the Input Link **Partitioning** tab.

   Note: You must perform the above modification rarely, and with extreme care, to avoid performance degradation.

3. Ensure that column metadata has been specified for the write.

Updating an IBM DB2 database

About this task

This is the same as writing to an IBM DB2 database, except you need to specify details of the SQL statements used to update the database:

- In the **Input Link Properties** tab:
  - Choose a Write Method of Upsert.
  - Choose the Upsert Mode, this allows you to specify whether to insert and update, or update only, and whether to use a statement automatically generated by InfoSphere DataStage or specify your own.
  - If you have chosen an Upsert Mode of User-defined Update and Insert, specify the Insert SQL statement to use. InfoSphere DataStage provides the auto-generated statement as a basis, which you can edit as required.
  - If you have chosen an Upsert Mode of User-defined Update and Insert or User-defined Update only, specify the Update SQL statement to use. InfoSphere DataStage provides the auto-generated statement as a basis, which you can edit as required.
  - If you want to send rejected rows down a rejects link, set Output Rejects to True (it is false by default).

Deleting rows from an IBM DB2 database

About this task

This is the same as writing an IBM DB2 database, except you need to specify details of the SQL statements used to delete rows from the database:

- In the **Input Link Properties** tab:
  - Choose a Write Method of Delete Rows.
  - Choose the Delete Rows Mode. This allows you to specify whether to use a statement automatically generated by the InfoSphere DataStage or specify your own.
  - If you have chosen a Delete Rows Mode of User-defined delete, specify the Delete SQL statement to use. InfoSphere DataStage provides the auto-generated statement as a basis, which you can edit as required.
  - If you want to send rejected rows down a rejects link, set Output Rejects to True (it is false by default).

Loading an IBM DB2 database

About this task

This is the default method. Note that loading is not supported by DB2 databases on a mainframe computer. Loading has the same requirements as writing, except:

- In the **Input Link Properties** tab:
  - Choose a Write Method of Load.

Reading data from an IBM DB2 database

About this task

To read an IBM DB2 database, follow the steps below.

Procedure

1. In the **Output Link Properties** tab:
   a. Choose a Read Method. This is Table by default, which reads directly from a table and operates in parallel mode. However, you can also choose to read using auto-generated SQL or user-generated SQL, which operates sequentially on a single node by default.
   b. Specify the table to be read.
   c. If using a Read Method of user-generated SQL, specify the SELECT SQL statement to use. InfoSphere DataStage provides the auto-generated statement as a basis, which you can edit as required.
   d. If using a Read Method apart from Table, you can specify a Partition Table property. This specifies execution of the query in parallel on the processing nodes containing a partition derived from the named table. If you do not specify this, the DB2/UDB enterprise stage executes the query sequentially on a single node.
   e. If you are not using environment variables to specify the server and database (as described in **Accessing IBM DB2 Databases** section), set Use Database Environment Variable and Use Server Environment Variable to False, and supply values for the Database and Server properties.
2. Ensure that column metadata has been specified for the Read.

Performing a direct lookup on an IBM DB2 database table

About this task

To perform a direct lookup on an IBM DB2 database, follow the steps below.

Procedure

1. Connect the DB2/UDB enterprise stage to a lookup stage by using a reference link.
2. In the **Output Link Properties** tab:
   a. Set the Lookup Type to Sparse.
   b. Choose a Read Method. This is Table by default, which reads directly from a table. However, you can also choose to read by using auto-generated SQL or user-generated SQL.
   c. Specify the table to be read for the lookup.
   d. If using a Read Method of user-generated SQL, specify the SELECT SQL statement to use. InfoSphere DataStage provides the auto-generated

statement as a basis, which you can edit as required. You would use this if, for example, you wanted to perform a non-equality based lookup.

e. If you are not using environment variables to specify the server and database (as described in the **Accessing IBM DB2 Databases** section), set Use Database Environment Variable and Use Server Environment Variable to False, and supply values for the Database and Server properties.

3. Ensure that column metadata has been specified for the lookup.

Performing an in-memory lookup on an IBM DB2 database table

About this task

In-memory lookup is the default lookup method. It has the same requirements as a direct lookup, except:

- In the **Output Link Properties** tab:
  - Set the Lookup Type to Normal.

Stage page

The **General** tab allows you to specify an optional description of the DB2/UDB enterprise stage. The **Advanced** tab allows you to specify how the stage executes. The **NLS Map** tab appears if you have NLS enabled on your system, it allows you to specify a character set map for the stage.

Advanced tab

This tab allows you to specify the following:

- **Execution Mode**. The stage can execute in parallel mode or sequential mode. In parallel mode the data is processed by the available nodes as specified in the Configuration file, and by any node constraints specified on the **Advanced** tab. In sequential mode the entire write is processed by the conductor node.
- **Combinability mode**. This is Auto by default, which allows the IBM InfoSphere DataStage to combine the operators that underlie parallel stages so that they run in the same process if it is sensible for this type of stage.
- **Preserve partitioning**. You can select **Set** or **Clear**. If you select **Set** file read operations will request that the next stage preserves the partitioning as is. The Preserve partitioning option does not appear if your stage only has an input link.
- **Node pool and resource constraints**. Select this option to constrain parallel execution to the node pool or pools and/or resource pool or pools specified in the grid. The grid allows you to make choices from drop down lists 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 open the **Available Nodes** dialog box and selecting nodes from there. You are effectively defining a new node pool for this stage (in addition to any node pools defined in the Configuration file).

  Note: The Stage page is blank if you are using the stage to perform a lookup directly on an IBM DB2 table, that is, operating in sparse mode.

NLS Map tab

The **NLS Map** tab allows you to define a character set map for the DB2/UDB enterprise stage. This overrides the default character set map set for the project or the job. You can specify that the map be supplied as a job parameter if required.

Input page

The Input page allows you to specify details about how the DB2/UDB enterprise stage writes data to an IBM DB2 database. The DB2/UDB enterprise stage can have only one input link writing to one table.

The **General** tab allows you to specify an optional description of the input link. The **Properties** tab allows you to specify details of exactly what the link does. The **Partitioning** tab allows you to specify how incoming data is partitioned before being written to the database. The **Columns** tab specifies the column definitions of incoming data. The **Advanced** tab allows you to change the default buffering settings for the input link.

Details about DB2/UDB enterprise stage properties, partitioning, and formatting are given in the following sections. See the topic about stage editors in the *IBM InfoSphere DataStage and QualityStage Parallel Job Developer's Guide* for a general description of the other tabs.

Input Link Properties tab

The **Properties** tab allows you to specify properties for the input link. These dictate how incoming data is written and where. Some of the properties are mandatory, although many have default settings. Properties without default settings appear in the warning color (red by default) and turn black when you supply a value for them.

The following table gives a quick reference list of the properties and their attributes.A more detailed description of each property follows.

Table 19. Input link properties and values

Category and Property	Values	Default	Required?	Dependent of
Target/Table	String	N/A	Yes	N/A
Target/Delete Rows Mode	Auto-generated delete/user-defined delete	Auto-generated delete	Yes if Write method = Delete Rows	N/A
Target/Delete SQL	String	N/A	Yes if Write method = Delete Rows	N/A
Target/Upsert Mode	Auto-generated Update & Insert/ Auto-generated Update Only/User-defined Update & Insert/User-defined Update Only	Auto-generated Update & Insert	Yes if Write method = Upsert	N/A
Target/Insert SQL	String	N/A	Yes if Write method = Upsert	N/A

Table 19. Input link properties and values (continued)

Category and Property	Values	Default	Required?	Dependent of
Target/Update SQL	String	N/A	Yes if Write method = Upsert	N/A
Target/Write Method	Delete Rows/Write/Load/ Upsert	Load	Yes	N/A
Target/Write Mode	Append/ Create/ Replace/ Truncate	Append	Yes	N/A
Connection/Use Default Database	True/False	True	Yes	N/A
Connection/Use Default Server	True/False	True	Yes	N/A
Connection/ Database	string	N/A	Yes (if Use Database environment variable = False)	N/A
Connection/ Server	string	N/A	Yes (if Use Server environment variable = False)	N/A
Connection/ Client Instance Name	string	N/A	No	N/A
Options/Array Size	number	2000	Yes (if Write Method = Delete)	N/A
Options/Output Rejects	True/False	False	Yes (if Write Method = Upsert)	N/A
Options/Row Commit Interval	number	value of Array Size	No	N/A
Options/Time Commit Interval	number	2	No	N/A
Options/Silently Drop Columns Not in Table	True/False	False	Yes	N/A
Options/ Truncate Column Names	True/False	False	Yes	N/A
Options/ Truncation Length	number	18	No	Truncate Column Names
Options/Close Command	string	N/A	No	N/A

Table 19. Input link properties and values (continued)

Category and Property	Values	Default	Required?	Dependent of
Options/Default String Length	number	32	No	N/A
Options/Open Command	string	N/A	No	N/A
Options/Use ASCII Delimited Format	True/False	False	Yes (if Write Method = Load)	N/A
Options/ Cleanup on Failure	True/False	False	Yes (if Write Method = Load)	N/A
Options/ Message File	path name	N/A	No	N/A
Options/DB Options	string	N/A	No	N/A
Options/ Nonrecoverable Transactions	True/False	False	No	N/A
Options/Pad Character	string	null	No	N/A
Options/ Exception Table	string	N/A	No	N/A
Options/ Statistics	stats_none/ stats_exttable_only/ stats_extindex_only/ stats_index/stats_table/ stats_extindex_table/ stats_all/ stats_both	stats_none	No	N/A
Options/ Number of Processes per Node	number	1	No	
Options/ Arbitrary Loading Order	True/False	True	No	Number of Processes per Node

Target category

Under **Target** category, you can set properties for the database table to write data to.

Table

Specify the name of the table to write to. You can specify a job parameter if required.

Delete Rows Mode

This option appears only for the Delete Rows write method, and allows you to specify how the delete statement is to be derived. Choose from:

- **Auto-generated Delete**. InfoSphere DataStage generates a delete statement for you, based on the values you have supplied for table name and column details. The statement can be viewed by selecting the **Delete SQL** property.
- **User-defined Delete**. Select this to enter your own delete statement. Then select the **Delete SQL** property and edit the default statement.

Delete SQL

Only appears for the Delete Rows write method. This property allows you to view an auto-generated Delete statement, or to specify your own, depending on the setting of the **Delete Rows Mode** property.

Upsert Mode

This only appears for the Upsert write method. Allows you to specify how the insert and update statements are to be derived. Choose from:

- **Auto-generated Update & Insert**. InfoSphere DataStage generates update and insert statements for you, based on the values you have supplied for table name and on column details. The statements can be viewed by selecting the **Insert SQL** or **Update SQL** properties.
- **Auto-generated Update Only**. InfoSphere DataStage generates an update statement for you, based on the values you have supplied for table name and on column details. The statement can be viewed by selecting the **Update SQL** properties.
- **User-defined Update & Insert**. Select this to enter your own update and insert statements. Then select the **Insert SQL** and **Update SQL** properties and edit the default statements.
- **User-defined Update Only**. Select this to enter your own update statement. Then select the **Update SQL** property and edit the default statement.

Insert SQL

Only appears for the Upsert write method. This property allows you to view an auto-generated Insert statement, or to specify your own (depending on the setting of the **Update Mode** property).

Update SQL

This property appears only for the Upsert write method. This property allows you to view an auto-generated Update statement, or to specify your own, depending on the setting of the **Update Mode** property.

Write Method

Choose from Delete Rows, Write, Upsert, or Load. Load is the default Write method. Load takes advantage of fast DB2 loader technology for writing data to the database. Note that loading is not supported by DB2 databases on a mainframe computer. Upsert uses Insert and Update SQL statements to write to the database.

Write Mode

Select from the following:

- **Append**. This is the default. New records are appended to an existing table.

- Create. Select this option to create a new table. If the IBM DB2 table already exists an error occurs and the job terminates. You must specify this mode if the IBM DB2 table does not exist.
- **Replace**. The existing table is first dropped and an entirely new table is created in its place. IBM DB2 uses the default partitioning method for the new table.

  Note that you cannot create or replace a table that has primary keys. You should not specify primary keys in your metadata.
- **Truncate**. The existing table attributes (including schema) and the IBM DB2 partitioning keys are retained, but any existing records are discarded. New records are then appended to the table.

Connection category

Under the **Connection** category, you can set properties for the database connection.

Use Default Server

This is set to True by default, which causes the DB2/UDB enterprise stage to use the setting of the DB2INSTANCE environment variable to derive the server. If you set this to False, you must specify a value for the Override Server property.

Use Default Database

This is set to True by default, which causes the stage to use the setting of the environment variable APT_DBNAME, if defined, and DB2DBDFT otherwise to derive the database. If you set the property to False, you must specify a value for the Override Database property.

Server

Optionally specify the IBM DB2 instance name for the table. This property appears if you set Use Server Environment Variable property to False.

Database

Optionally specify the name of the IBM DB2 database to access. This property appears if you set Use Database Environment Variable property to False.

Client Instance Name

This property is required only if you are connecting to a remote IBM DB2 server. It specifies the IBM DB2 client through which you are making the connection (see the **Remote connection** section).

Note: Connection details are normally specified by environment variables as described in the **Accessing IBM DB2 databases** section. If you are specifying a remote connection, when you fill in the client instance name, user and password fields appear and allow you to specify these for connection to the remote server.

Options category

Under the **Options** category, you can set additional properties for the job that you are creating.

Array Size

This is only available for Write Methods of Delete and Upsert, and is optional for Upsert. You specify the size the insert/delete host array. It defaults to 2000, but you can enter 1 if you want each insert/delete statement to be executed individually.

Output Rejects

This appears for the Upsert Write Method. It specifies how to handle rows that fail to be inserted. Choose True to send them down a reject link, or False to drop them. A state field is added to each rejected row. This field contains a five-letter SQL code that identifies the reason that the record was rejected.

Row Commit Interval

This is available for Write Methods of Upsert, Delete Rows, and Write. It specifies the number of records that should be committed before starting a new transaction. The specified number must be a multiple of the array size. For Upsert and Delete Rows, the default is the array size (which in turn defaults to 2000). For Write the default is 2000.

If you set a small value for Row Commit Interval, you force IBM DB2 to perform frequent commits. Therefore, if your program terminates unexpectedly, your data set can still contain partial results that you can use. However, you might pay a performance penalty because of the high frequency of the commits. If you set a large value for Row Commit Interval, IBM DB2 must log a correspondingly large amount of rollback information. This, too, might slow your application.

Time Commit Interval

This is available for Write Methods of Upsert and Delete. It specifies the number of seconds InfoSphere DataStage should allow between committing the input array and starting a new transaction. The default time period is 2 seconds.

Silently Drop Columns Not in Table

This is False by default. Set to True to silently drop all input columns that do not correspond to columns in an existing IBM DB2 table. Otherwise the stage reports an error and terminates the job.

Truncate Column Names

Select this option to truncate column names to 18 characters. To specify a length other than 18, use the Truncation Length dependent property:

- **Truncation Length**

  This is set to 18 by default. Change it to specify a different truncation length.

Close Command

This is an optional property. Use it to specify any command to be parsed and executed by the IBM DB2 database on all processing nodes after the stage finishes processing the IBM DB2 table. You can specify a job parameter if required.

Default String Length

This is an optional property and is set to 32 by default. Sets the default string length of variable-length strings written to an IBM DB2 table. Variable-length strings longer than the set length cause an error.

The maximum length you can set is 4000 bytes. Note that the stage always allocates the specified number of bytes for a variable-length string. In this case, setting a value of 4000 allocates 4000 bytes for every string. Therefore, you should set the expected maximum length of your largest string and no larger.

Open Command

This is an optional property. Use it to specify any command to be parsed and executed by the IBM DB2 database on all processing nodes before the IBM DB2 table is opened. You can specify a job parameter if required.

Use ASCII Delimited Format

This property only appears if Write Method is set to Load. Specify this option to configure IBM DB2 to use the ASCII-delimited format for loading binary numeric data instead of the default ASCII-fixed format.

This option can be useful when you have variable-length columns, because the database will not have to allocate the maximum amount of storage for each variable-length column. However, all numeric columns are converted to an ASCII format by IBM DB2, which is a CPU-intensive operation. See the IBM DB2 reference manuals for more information.

Cleanup on Failure

This property appears only if Write Method is set to Load. Specify this option to deal with failures during stage execution that leave the table space being loaded in an inaccessible state.

The cleanup procedure neither inserts data into the table nor deletes data from it. You must delete rows that were inserted by the failed execution either through the IBM DB2 command-level interpreter or by using the stage subsequently using the **replace** or **truncate write** modes.

Message File

This property only appears if Write Method is set to Load. Specify the file where the IBM DB2 loader writes diagnostic messages. The database instance must have read/write privilege to the file.

DB Options

This appears only if Write Method is set to load and Write Mode is set to Create or Replace. Specify an optional table space or partitioning key to be used by IBM DB2 to create the table.

By default, the InfoSphere DataStage creates the table on all processing nodes in the default table space and uses the first column in the table, corresponding to the first field in the input data set, as the partitioning key.

You specify arguments as a string enclosed in braces in the form:
`{tablespace=t_space,[key=col0,...]}`

Nonrecoverable Transactions

This option appears only if Write Method is set to Load. It is False by default. If set to True, it indicates that your load transaction is marked as nonrecoverable. It will not be possible to recover your transaction with a subsequent roll forward action. The roll forward utility will skip the transaction, and will mark the table into which data was being loaded as "invalid". The utility will also ignore any subsequent transactions against the table. After a roll forward is completed, the table can only be dropped. Table spaces are not put in a backup pending state following the load operation, and a copy of the loaded data is not made during the load operation.

Pad Character

This appears for a Write Method of Upsert or Delete Rows. It specifies the padding character to be used in the construction of a WHERE clause when it contains string columns that have a length less than the DB2 char column in the database. It defaults to null. (See the section titled **Using the Pad Character property**)

Exception Table

This property appears only if Write Method is set to Load. It allows you to specify the name of a table where rows that violate load table constraints are inserted. The table needs to have been created in the IBM DB2 database. The exception table cannot be used when the Write Mode is set to create or replace.

Statistics

This property appears only if Write Method is set to Load. It allows you to specify which statistics should be generated upon load completion, as part of the loading process IBM DB2 will collect the requisite statistics for table optimization. This option is only valid for a Write Mode of truncate, it is ignored otherwise.

Number of Processes per Node

This property only appears if Write Method is set to Load. It allows you to specify the number of processes to initiate on every node. If set to 0, the stage uses its own algorithm to determine the optimal number, based on the number of CPUs available at runtime. This does not, however, take into account the workload from the rest of the job. By default it is set to 1. It has the following dependent property:

- **Arbitrary Loading Order**

  This only appears if Number of Processes per Node is set to a value greater than 1. If set true, it specifies that the loading of every node can be arbitrary, leading to a potential performance gain.

Connection category

Under the **Connection** category for the **Input** link, you can set appropriate properties for the database connection.

Use Default Database

This is set to True by default, which causes the stage to use the default IBM DB2 subsystem. If you set the property to False, you must specify a value for the Override Database property.

Database

Optionally specify the name of the IBM DB2 database to access. This property appears if you set Use Database Environment Variable property to False.

MVS DataSets category

Under this category, you can specify appropriate properties for MVS data sets.

Discard DSN

Specify the name of the MVS data set that stores the rejected records. It has the following subproperties:

- **Discard Device Type**

  The device type that is used for the specified discard data set.
- **Discard Space**

  The primary allocation space for the discard data set, specified in cylinders.
- **Max Discards Per Node**

  An integer which specifies the maximum number of discarded rows to keep in a data set per node.

Error DSN

The name of the MVS data set that stores rows that could not be loaded into IBM DB2 because of an error. It has the following subproperties:

- **Error Device Type**

  The device type that is used for the specified Error data set.
- **Error Space**

  The primary allocation space for the error data set, specified in cylinders.

Map DSN

Specify the name of the MVS data set for mapping identifiers back to the input records that caused an error. This property has the following subproperties:

- **Map Device Type**

  The device type that is used for the specified Map data set.
- **Map Space**

  The primary allocation space for the map data set, specified in cylinders.

Work 1 DSN

Specify the name of the MVS data set for sorting input. This property has the following subproperties:

- **Work 1 Device Type**

  The device type that is used for the specified Work 1 data set.
- **Work 1 Space**

  The primary allocation space for the Work 1 data set, specified in cylinders.

Work 2 DSN

Specify the name of the MVS data set for sorting output. This property has the following subproperties:

- **Work 2 Device Type**
  The device type that is used for the specified Work 2 data set.
- **Work 2 Space**
  The primary allocation space for the Work 2 data set, specified in cylinders.

Options category

Under this category, you can specify additional properties for the write operation.

Enforce Constraints

This property is available only when Write Method is set to Load. If this is set to True, load will delete errant rows when encountering them, and issue a message identifying such row. This requires that:

- referential constraints exist.
- the input must be sorted.
- a Map DSN data set must be specified under the MVS data sets category.

Keep Dictionary

This property is available only when Write Method is set to Load. If this is set to true, Load is prevented from building a new compression dictionary. This property is ignored unless the associated table space has the COMPRESS YES attribute.

Preformat

This property is available only when Write Method is set to Load. If set to True, the remaining pages are pre-formatted in the table space and its index space.

Silently Drop Columns Not in Table

This is False by default. Set to True to silently drop all input columns that do not correspond to columns in an existing IBM DB2 table. Otherwise the stage reports an error and terminates the job.

Truncate Column Names

Select this option to truncate column names to 18 characters. To specify a length other than 18, use the Truncation Length dependent property:

- **Truncation Length**
  This is set to 18 by default. Change it to specify a different truncation length.

Verbose

This option is available only when Write Method is set to Load. If this is set to True, the InfoSphere DataStage logs all messages generated by IBM DB2 when a record is rejected because of prime key or other violations.

Close Command

This is an optional property. Use it to specify any command to be parsed and executed by the IBM DB2 database on all processing nodes after the stage finishes processing the IBM DB2 table. You can specify a job parameter if required.

Default String Length

This is an optional property and is set to 32 by default. Sets the default string length of variable-length strings written to an IBM DB2 table. Variable-length strings longer than the set length cause an error.

The maximum length you can set is 4000 bytes. Note that the stage always allocates the specified number of bytes for a variable-length string. In this case, setting a value of 4000 allocates 4000 bytes for every string. Therefore, you should set the expected maximum length of your largest string and no larger.

Exception Table

This property only appears if Write Method is set to Load. It allows you to specify the name of a table where rows that violate load table constraints are inserted. The table needs to have been created in the IBM DB2 database. The exception table cannot be used when the Write Mode is set to create or replace.

Number of Processes per Node

This property only appears if Write Method is set to Load. It allows you to specify the number of processes to initiate on every node. If set to 0, the stage uses its own algorithm to determine the optimal number, based on the number of CPUs available at runtime. This does not, however, take into account the workload from the rest of the job. By default it is set to 1. This property has the following dependent property:

- **Arbitrary Loading Order**

  This only appears if Number of Processes per Node is set to a value greater than 1. If set true, it specifies that the loading of every node can be arbitrary, leading to a potential performance gain.

Open Command

This is an optional property. Use it to specify any command to be parsed and executed by the IBM DB2 database on all processing nodes before the IBM DB2 table is opened. You can specify a job parameter if required.

Row Estimate

This option is available only when Write Method is set to Load. Specify the estimated number of rows (across all nodes) to be loaded into the database. An estimate of the required primary allocation space for storing all rows is made before load is engaged.

Sort Device Type

This option is available only when Write Method is set to Load. Specify the device type for dynamically allocated data sets used by DFSORT.

Sort Keys

This option is available only when Write Method is set to Load. Set this to have rows presorted according to keys, the value is an estimate of the number of index keys to be sorted. Do not use this property if table space does not have an index, has only one index, or data is already sorted according to index keys.

When Clause

This option is available only when Write Method is set to Load. Specify a WHEN clause for the load script.

Create Statement

This option is available only when Write Method is set to Load and Write Mode is set to Create or Replace. Specify the SQL statement to create the table.

DB Options

This option appears only if Write Method is set to load and Write Mode is set to Create or Replace. Specify an optional table space or partitioning key to be used by IBM DB2 to create the table.

By default, the InfoSphere DataStage creates the table on all processing nodes in the default table space and uses the first column in the table, corresponding to the first field in the input data set, as the partitioning key.

You specify arguments as a string enclosed in braces in the form:

`{tablespace=`*`t_space`*`,[key=`*`col0`*`,...]}`

Reuse Datasets

This option appears only if Write Method is set to Load and Write Mode is set to Replace. If True, IBM DB2 reuses IBM DB2 managed data sets without relocating them.

Statistics

This only appears if Write Method is set to load and Write Mode is set to Truncate. Specify which statistics should be generated upon completion of load. As a part of the loading process, IBM DB2 collects the statistics required for table access optimization. Alternatively, use the RUNSTAT utility.

Array Size

This option is available only for Write Methods of Delete and Upsert, and is optional for upsert. This specifies the size of the insert/delete host array. It defaults to 2000, but you can enter 1 if you want each insert/delete statement to be executed individually.

Pad Character

This option appears for a Write Method of Upsert or Delete Rows. It specifies the padding character to be used in the construction of a WHERE clause when it contains string columns that have a length less than the DB2 char column in the

database. It defaults to null. (See the section titled **Using the Pad Character property**.)

Row Commit Interval

This option is available for Write Methods of Upsert, Delete Rows, and Write. Specify the number of records that should be committed before starting a new transaction. The specified number must be a multiple of the array size. For Upsert and Delete Rows, the default is the array size (which in turn defaults to 2000). For Write the default is 2000.

If you set a small value for Row Commit Interval, you force IBM DB2 to perform frequent commits. Therefore, if your program terminates unexpectedly, your data set can still contain partial results that you can use. However, you might pay a performance penalty because of the high frequency of the commits. If you set a large value for Row Commit Interval, IBM DB2 must log a correspondingly large amount of rollback information. This, too, might slow your application.

Time Commit Interval

This option is available for Write Methods of Upsert and Delete. Specify the number of seconds that the InfoSphere DataStage should allow between committing the input array and starting a new transaction. The default time period is 2 seconds.

Output Rejects

This appears for the Upsert Write Method. It specifies how to handle rows that fail to be inserted. Choose True to send them down a reject link, or False to drop them. A state field is added to each rejected row. This field contains a five-letter SQL code that identifies the reason that the record was rejected.

Partitioning tab

The **Partitioning** tab allows you to specify details about how the incoming data is partitioned or collected before it is written to the IBM DB2 database. It also allows you to specify that the data should be sorted before being written.

By default the stage partitions in DB2 mode. This takes the partitioning method from a selected IBM DB2 database (or the one specified by the environment variables described in the **Accessing IBM DB2 Databases** section.)

If the DB2/UDB enterprise stage is operating in sequential mode, it will first collect the data before writing it to the file using the default Auto collection method.

The **Partitioning** tab allows you to override this default behavior. The exact operation of this tab depends on:

- Whether the DB2/UDB enterprise stage is set to execute in parallel or sequential mode.
- Whether the preceding stage in the job is set to execute in parallel or sequential mode.

If the DB2/UDB enterprise stage is set to execute in parallel, then you can set a partitioning method by selecting from the **Partition type** list. This will override any current partitioning.

If the DB2/UDB enterprise stage is set to execute in sequential mode, but the preceding stage is executing in parallel mode, then you can set a collection method from the **Collector type** list. This will override the default Auto collection method.

The following partitioning methods are available:

- **Entire**. Each file to which data is written, receives the entire data set.
- Hash. The records are hashed into partitions based on the value of a key column or columns selected from the **Available** list.
- **Modulus**. The records are partitioned using a modulus function on the key column selected from the **Available** list. This is commonly used to partition on tag columns.
- **Random**. The records are partitioned randomly, based on the output of a random number generator.
- **Round Robin**. The records are partitioned on a round robin basis as they enter the stage.
- **Same**. Preserves the partitioning already in place.
- **DB2**. Replicates the IBM DB2 partitioning method of the specified IBM DB2 table. This is the default method for the DB2/UDB enterprise stage.
- **Range**. Divides a data set into approximately equal size partitions based on one or more partitioning keys. Range partitioning is often a preprocessing step to performing a total sort on a data set. Requires extra properties to be set. Access these properties by clicking the properties button.

The following Collection methods are available:

- **(Auto)**. This is the default collection method for DB2/UDB enterprise stage. Normally, when you are using the Auto mode, the InfoSphere DataStage will read any row from any input partition as it becomes available.
- **Ordered**. Reads all records from the first partition, then all records from the second partition, and so on.
- **Round Robin**. Reads a record from the first input partition, then from the second partition, and so on. After reaching the last partition, the operator starts over.
- **Sort Merge**. Reads records in an order based on one or more columns of the record. This requires you to select a collecting key column from the **Available** list.

The **Partitioning** tab also allows you to specify that data arriving on the input link should be sorted before being written to the file or files. The sort is always carried out within data partitions. If the stage is partitioning incoming data the sort occurs after the partitioning. If the stage is collecting data, the sort occurs before the collection. The availability of sorting depends on the partitioning or collecting method chosen. It is not available with the default Auto methods.

Select the check boxes as follows:

- **Perform Sort**. Select this to specify that data coming in on the link should be sorted. Select the column or columns to sort on from the **Available** list.
- **Stable**. Select this if you want to preserve previously sorted data sets. This is the default.
- **Unique**. Select this to specify that, if multiple records have identical sorting key values, only one record is retained. If stable sort is also set, the first record is retained.

If NLS is enabled an additional button opens a dialog box allowing you to select a locale specifying the collate convention for the sort.

You can also specify sort direction, case sensitivity, whether sorted as ASCII or EBCDIC, and whether null columns will appear first or last for each column. Where you are using a keyed partitioning method, you can also specify whether the column is used as a key for sorting, for partitioning, or for both. Select the column in the **Selected** list and right-click to invoke the shortcut menu.

Output page

The Output page allows you to specify details about how the DB2/UDB enterprise stage reads data from an IBM DB2 database. The stage can have only one output link. Alternatively, it can have a reference output link, which is used by the Lookup stage when referring to a IBM DB2 lookup table. It can also have a reject link where rejected records are routed (used in conjunction with an input link).

The **General** tab allows you to specify an optional description of the output link. The **Properties** tab allows you to specify details of exactly what the link does. The **Columns** tab specifies the column definitions of the data. The **Advanced** tab allows you to change the default buffering settings for the output link.

Details about DB2/UDB enterprise stage properties are given in the following sections. See the *IBM InfoSphere DataStage and QualityStage Parallel Job Developer's Guide* for a general description of the other tabs.

Output Link Properties tab

The **Properties** tab allows you to specify properties for the output link. These properties dictate how incoming data is read from what table. Some of the properties are mandatory, although many have default settings. Properties without default settings appear in the warning color (red by default) and turn black when you supply a value for them.

The **Build SQL** button allows you to instantly open the SQL Builder to help you construct an SQL query to read data. See the topic about SQL Builder in the *IBM InfoSphere DataStage and QualityStage Designer Client Guide* for guidance on using it.

The following table gives a quick reference list of the properties and their attributes. A more detailed description of each property follows.

Table 20. Output link properties and values

Category and Property	Values	Default	Required?	Dependent of
Source/Lookup Type	Normal/ Sparse	Normal	Yes (if output is reference link connected to Lookup stage)	N/A
Source/Read Method	Table/ Auto-generated SQL/User-defined SQL	Table	Yes	N/A
Source/Table	string	N/A	Yes (if Read Method = Table)	N/A

Table 20. Output link properties and values (continued)

Category and Property	Values	Default	Required?	Dependent of
Source/Where clause	string	N/A	No	Table
Source/Select List	string	N/A	No	Table
Source/Query	string	N/A	Yes (if Read Method = Query)	N/A
Source/Partition Table	string	N/A	No	Query
Connection/Use Default Database	True/False	True	Yes	N/A
Connection/Use Default Server	True/False	True	Yes	N/A
Connection/ Server	string	N/A	Yes (if Use Database environment variable = False)	N/A
Connection/ Database	string	N/A	Yes (if Use Server environment variable = False)	N/A
Connection/ Client Instance Name	string	N/A	No	N/A
Options/Close Command	string	N/A	No	N/A
Options/Open Command	string	N/A	No	N/A

Source category

Under the **Source** category, you can specify the properties of the database to read data from.

Lookup Type

Where the DB2/UDB enterprise stage is connected to a Lookup stage via a reference link, this property specifies whether the stage will provide data for an in-memory look up (Lookup Type = Normal) or whether the lookup will access the database directly (Lookup Type = Sparse). If the Lookup Type is Normal, the Lookup stage can have multiple reference links. If the Lookup Type is Sparse, the Lookup stage can only have one reference link.

Read Method

This property specifies whether you are specifying a table or a query when reading the IBM DB2 database, and how you are generating the query:

- Select the Table method in order to use the Table property to specify the read. This will read in parallel mode.

- Select Auto-generated SQL to have the InfoSphere DataStage automatically generate an SQL query based on the columns you have defined and the table you specify in the Table property.
- Select User-defined SQL to define your own query.
- Select SQL Builder Generated SQL to open the SQL Builder and define the query using its helpful interface. (See the topic about SQL Builder in the *IBM InfoSphere DataStage and QualityStage Designer Client Guide*.

By default, Read methods of SQL Builder Generated SQL, Auto-generated SQL, and User-defined SQL operate sequentially on a single node. You can have the User-defined SQL read operate in parallel if you specify the Partition Table property.

Query

This property is used to contain the SQL query when you choose a Read Method of User-defined query or Auto-generated SQL. If you are using Auto-generated SQL you must select a table and specify some column definitions. An SQL statement can contain joins, views, database links, synonyms, and so on. It has the following dependent option:

- **Partition Table**

  Specifies execution of the query in parallel on the processing nodes containing a partition derived from the named table. If you do not specify this, the stage executes the query sequentially on a single node.

Table

Specifies the name of the IBM DB2 table. The table must exist and you must have SELECT privileges on the table. If your IBM DB2 user name does not correspond to the owner of the specified table, you can prefix it with a table owner in the form:

```
table_owner.table_name
```

If you use a Read method of Table, then the Table property has two dependent properties:

- **Where clause**

  Allows you to specify a WHERE clause of the SELECT statement to specify the rows of the table to include or exclude from the read operation. If you do not supply a WHERE clause, all rows are read.
- **Select List**

  Allows you to specify an SQL select list of column names.

Connection category

Under this category, you can set appropriate properties for the server and database to connect to.

Use Default Server

This is set to True by default, which causes the stage to use the setting of the DB2INSTANCE environment variable to derive the server. If you set this to False, you must specify a value for the Override Server property.

Use Default Database

This is set to True by default, which causes the stage to use the setting of the environment variable APT_DBNAME, if defined, and DB2DBDFT otherwise to derive the database. If you set the property to False, you must specify a value for the Override Database property.

Server

Optionally specify the IBM DB2 instance name for the table. This property appears if you set the Use Server Environment Variable property to False.

Database

Optionally specify the name of the IBM DB2 database to access. This property appears if you set Use Database Environment Variable property to False.

Client Instance Name

This property is required only if you are connecting to a remote IBM DB2 server. It specifies the IBM DB2 client through which you are making the connection (see the **Remote connection** section).

Note: Connection details are normally specified by environment variables as described in the **Accessing IBM DB2 Databases** section. If you are specifying a remote connection, when you fill in the client instance name, user and password fields appear and allows you to specify these for connection to the remote server.

Options category

Under this category, you can specify the close and open SQL queries, and the Pad character.

Close Command

This is an optional property. Use it to specify a command to be parsed and executed by the IBM DB2 database on all processing nodes after the stage finishes processing the IBM DB2 table. You can specify a job parameter if required.

Open Command

This is an optional property. Use it to specify a command to be parsed and executed by the IBM DB2 database on all processing nodes before the IBM DB2 table is opened. You can specify a job parameter if required.

Pad Character

This appears when you are using an IBM DB2 table as a lookup, that is, when you have set Lookup Type as Sparse. It specifies the padding character to be used in the construction of a WHERE clause when it contains string columns that have a length less than the DB2 char column in the database. It defaults to null. (See the section titled **Using the Pad Character property**.)

Chapter 6. DB2 UDB Load stage

Use the IBM DB2 UDB Load stage to write data to a DB2 database from an IBM InfoSphere DataStage server or parallel job.

When you use IBM InfoSphere DataStage to write DB2 data, you can choose from a collection of connectivity options. For most new jobs, use the DB2 Connector stage, which offers better functionality and performance than the DB2 UDB Load stage.

If you have jobs that use the DB2 UDB Load stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

Functionality of DB2 UDB Load stage

Some of the functionality of DB2 UDB Load stage are: support of the Sequential File and the Named Pipe methods for loading data and support of load parameters to control the load process.

The DB2 UDB Load stage has the following functionality:

- Support for NLS (National Language Support).
- Bulk loading from a stream input link to provide rows of data into the target table.
- Support of the Sequential File and the Named Pipe methods for loading data.
- Support of INSERT, REPLACE, and RESTART load modes.
- Support of load parameters to control the load process.
- Support for data files that exceed the 2 GB file size limit for 64 bit file systems.

Setting environment variables for IBM DB2 Database

You must set certain environment variables in order for the stage to work correctly on a UNIX platform.

About this task

Set the following environment variables on the server machine.

- DB2INSTANCE
- INSTHOME
- LD_LIBRARY_PATH

The actual name of the environment variable LD_LIBRARY_PATH differs depending on the platform.

- If the platform is AIX, use LIBPATH.
- If the platform is HP_UX, use SHLIB_PATH.
- If the platform is LINUX or Solaris, use LD_LIBRARY_PATH.

Load Methods

The two methods of loading data into an IBM DB2 table are the Sequential File method and the Named Pipe method. The **Load Method** property determines which method to use to load the data.

Sequential File Method

When you load data by using the Sequential File method, rows from the input link are written in delimited format into a sequential file called a data file.

Depending on the value of the Load Immediate property, the data file is loaded immediately or the load is delayed. Sequential File loading is slower than the Named Pipe loading because all the rows must be written to this data file.

- In immediate loading, a data file (INPDATA.DAT) is constructed, which contains the rows of data to be loaded.
- In delayed loading, the following three files are constructed:
  - **INPDATA.DAT.** The data file containing the rows of data to be loaded.
  - **CMD.CLP.** The command file containing the Connect, Load, and Quit commands. The Load command is constructed from property values.
  - **ULOAD.BAT.** The batch file that calls the command file. The data file is loaded by running this custom batch file.

    The advantage of using delayed loading is that you can modify the data file and command file or move them to another machine.

Named Pipe Method

When you load data by using the Named Pipe method, the rows from the input link are streamed continuously into the named pipe for loading the data until the end of the data.

Use this method when you need to load the rows from the input link immediately as they are streamed into the pipe.

Restarting the Load

While restarting a load that failed in the build or delete phases, there are some requirements that should be considered. The BUILD, DELETE, and LOAD values refer to the Restart Phase property where as the INSERT, REPLACE, and RESTART values refer to the Load Mode property.

About this task

To restart a load that failed in the build or delete phases, consider the following requirements:

- Use the same parameters for the restart phase as you previously specified for the interrupted load.
- RESTART does not reread fresh data from the data file or the named pipe. It builds indexes during the build restart phase and deletes all rows. This causes a unique key violation during the delete restart phase. Therefore, do not use RESTART to run a job in the build or delete phases. (Rows from the input link would be unnecessarily read.)

To restart the load that failed in INSERT or REPLACE mode, use one of the following options:

- **Modify the command file.** Run the *uload.bat* batch file that resides in the directory specified in the Directory for Data and Command Files property. The *uload.bat* file runs a command file that executes the IBM DB2 LOAD command.

  You must then change the LOAD command in the *cmd.clp* command file from INSERT (or REPLACE) to RESTART and set RESTARTCOUNT to B (build) or D (delete). Use this method when you want to restart the load without any of the InfoSphere DataStage processing.
- **Change the property values.** Change the Load Mode value to RESTART and the restart phase value to LOAD, BUILD, or DELETE.

  All other property values should remain the same as specified in the previously interrupted load. When you use this option, you must wait until all the rows are read from the previous stages. Because this is time-consuming, use this method only if you are unfamiliar with the LOAD command syntax.

Loading an IBM DB2 Database

Load an IBM DB2 database using the appropriate loading method.

About this task

Use the InfoSphere DataStage and QualityStage Designer to complete the following procedure.

Procedure

1. Add a DB2 UDB Load stage to your InfoSphere DataStage job.
2. Link the DB2 UDB Load stage to its data source.
3. Specify column definitions by using the **Columns** tab.
4. Determine the appropriate loading method, as documented in "Load Methods".
5. Add the appropriate property values on the **Stage** tab.
6. Compile the job.
7. If the job compiles correctly, you can choose one of the following:
   - Run the job from within the Designer client
   - Run or schedule the job by using the InfoSphere DataStage and QualityStage Director
8. If the job does not compile correctly, correct the errors and recompile.

DB2 UDB Load stage - Properties tab

Use the **Properties** tab to specify details about the load operation.

The **Properties** tab contains the following properties:

Table 21. DB2 UDB Load stage properties

Prompt	Type	Default	Description
Database Name	String	None	The name of the database containing the table to be loaded. This database alias name is specified in the client configuration setup.
User Name	String	None	The user name used to log on to the specified database.
Password	String	None	The password to use for the specified user.

Table 21. DB2 UDB Load stage properties (continued)

Prompt	Type	Default	Description
Table Name	String	None	The name of the table into which the data is loaded. The table name must be prefixed by its schema name if the schema name is different from the value specified in User Name.
Load Method	String List	Named Pipe	The Named Pipe or Sequential File method used to load the data into the target table. (Named Pipe or Sequential File)
Load Immediate	String List	Yes	Specifies whether to load the data immediately or to construct a data file and a batch file for loading the data later. Use one of the following load methods: **Named Pipe load method.** Supports immediate loading. **File load method.** Supports immediate and delayed loading. In delayed loading, data, command, and batch files are constructed. The data file is later loaded by running the batch file. If set to **Yes**, the load is immediate. If set to **No**, the load is delayed. This property is valid only when **Load Method** is set to **Sequential File**. (Yes or No)
Load from client	String List	No	Indicates whether to allow loading of data from a remotely connected client. Use one of the following values: **No** - The load cannot be done from a remote client. **Yes** - The load can be done from a remote client. This option is available at IBM DB2 Server version 7.1 or later. (No or Yes)
Directory for Data and Command Files	String	None	The directory in which the data file (INPDATA.DAT), command file (CMD.CLP), and batch files (ULOAD.BAT) are generated. If these files already exist, they are overwritten. If you specify the load as Sequential File (delayed), all three files are generated. If the load is immediate and an error occurs when loading data, the command file, batch file (and a data file if **Remove Intermediate Datafile** is set to **No**) are generated. You can modify these files if necessary and use them to restart the load.
Remove Intermediate Datafile	String List	Yes	Specifies whether to remove the data file after loading it if you specify the load as Sequential File (immediate). If set to **Yes**, the data file is deleted. If set to **No**, the data file is retained. (Yes or No)

Table 21. DB2 UDB Load stage properties (continued)

Prompt	Type	Default	Description
File type of the data format	String List	DEL	The format of the data: **ASC** - Non-delimited ASCII format. **DEL** - Delimited ASCII format. **IXF** - (Integrated exchange format, PC version) exported from the same or from another IBM DB2 table. **CURSOR** - A cursor declared against a SELECT of VALUES statement. • ASC • DEL • IXF • CURSOR
LOB path	String	None	The path to the data files that contains LOB values to be loaded. Use a comma (,) to separate paths.
File type modifier	String	lobsinfile noheader	Specifies additional options used by MODIFIED BY. For list and description of options, see IBM DB2 documentation.
Method	String List	D	Specifies how to load data into columns in a table. **L** - Specifies the start and end column numbers from which to load data. **N** - Specifies the names of the columns in the data file to be loaded. **P**- Specifies the field numbers of the input data fields to be loaded. **D** - Specifies Default. • L • N • P • D
Column-start Column-end	String	None	The start and end column numbers from which to load data. A column number is a byte offset from the beginning of a row of data. It is numbered starting from 1. This property is valid only when **Method** is set to **L**.
Column name	String	None	The names of the columns in the data file to be loaded. The case of these column names must match the case of the corresponding names in the system catalogs. This property is valid only when **Method** is set to **N**.
Column position	String	None	The field numbers (numbered from 1) of the input data files to be loaded. This property is valid only when **Method** is set to **P**.
Null Indicators	String	None	A comma-separated list of positive integers specifying the column number of each null indicator field. This property is valid only when **Method** is set to **L**.

Table 21. DB2 UDB Load stage properties (continued)

Prompt	Type	Default	Description
Rows Buffer Size	Long	4	The number in KB specifying the size of the buffer for the rows from the input link. The rows are buffered before loading them into either the sequential file or a named pipe.
Load Mode*	String List	INSERT	The mode in which the load takes place. Use one of the following modes: **INSERT.** Adds the loaded data to the table without changing the existing table data. **REPLACE.** Deletes all existing data from the table, and inserts the loaded data. **RESTART.** Restarts the load after a previous load was interrupted. **TERMINATE.** Terminates the previously interrupted load and moves the table spaces in which the table resides from the load pending state to the recovery pending state. • INSERT • REPLACE • RESTART • TERMINATE
Save Count*	Long	0	The number of records to load before establishing a consistency point. The default value 0 means no consistency points are established unless necessary.
Row Count*	Long	0	The number of physical records to load. You can load only the first *n* rows in a file. The default value 0 means load all rows.
Restart Count*	Long	0	The number of records to skip before starting to load records. Use this property if a previous attempt to load records failed after some records were committed to the database. The default value 0 means start the load from row 1.
Restart Phase*	String List	None	The phase at which to restart the load operation. You can restart the load at the load, build, or delete phases. Do not specify restart phase values BUILD or DELETE for INSERT or REPLACE load modes. • LOAD • BUILD • DELETE
Warning Count*	Long	0	The number of warnings after which the load must be stopped. The default value 0 means the load continues regardless of the number of warnings issued.
Local Message File Name*	String	None	The string containing the path name for the local file name used for output messages.

Table 21. DB2 UDB Load stage properties (continued)

Prompt	Type	Default	Description
Directory for temporary files*	String	None	The string containing the path name for the file name used by database server for temporary files. Temporary files are created to store messages and consistency points and to delete phase information. You must ensure that each load has a unique fully qualified remote file name.
Insert-column	String	None	Specifies the table column into which the data is to be inserted.
Datalink specification	String	None	Specifies DATALINK columns. For each DATALINK column, there can be only one column specification enclosed by parenthesis. Each column specification consists of one or more DL_LINKTYPE, prefix, and a DL_URL_SUFFIX specification.
Exception Table Name*	String	None	(Optional) The exception table name into which rows in error are copied during the load. This table should exist in the database at the time of loading.
Statistics*	String List	NoStatistics	Indicates the type of statistics to gather for the table. Collection of statistics is not supported for INSERT or RESTART load modes. The default value NoStatistics means no statistics are gathered for the table. • TableStats • TableAndIndexesStats • IndexStats • TableAndDistributedStats • TableAndDistributedStats • AndBasicIndexes • ExtendedStatsForIndexOnly • ExtendedStatsForIndexes • AndBasicTableStats • AllStatistics • NoStatistics
Non Recoverable*	String List	No	Indicates whether to mark the load transaction as nonrecoverable. Thus, it is not possible to recover it by a subsequent roll-forward action. The default value **No** means the load transaction is to be marked recoverable. (Yes/No)

Table 21. DB2 UDB Load stage properties (continued)

Prompt	Type	Default	Description
Without prompting	String List	NO	Specifies that the list of data files contains all the files that are to be loaded, and that the devices or directories listed are sufficient for the entire load operation. **YES** - Indicates the list of data files and the devices or directories listed are sufficient for the entire load operation. If a continuation input files is not found, the load fails, and the table remains in a load-pending state. **NO** - Indicates the option is not specified. If the tape device encounters an end of tape for the copy image or the last item listed is a tape device, the user is prompted for a new tape on that device. (YES/NO)
Data Buffer Size*	Long	0	The number specifying how many 4-KB pages to use as buffered space for transferring data within the load utility. The default value 0 means the load utility calculates an intelligent value at run time.
Sort Buffer Size*	Long	0	The number of 4-KB pages of memory to use for sorting the index keys during a load operation. Sort buffer size greatly impacts sort performance. Therefore, for very large tables (for example, tables in excess of 100 MB, set this buffer as large as possible.) The default value 0 means the load utility calculates the value at run time.
Working Directory*	String	None	The optional working directory used for sorting index keys. If no value exists, the *sqllib/tmp* directory in the database server installation directory is used.
CPU Parallelism*	Long	0	The number of processes or threads that the load utility should spawn for parsing, converting, and formatting records when building table objects. The default value 0 means the load utility chooses a value at run time.
Disk Parallelism*	Long	0	The number of processes or threads that the load utility should spawn for writing data to the table space containers. The default value 0 means the load utility chooses a value based on the number of table space containers and the characteristics of the table.
Indexing Mode	String List	AUTOSELECT	Specifies whether the load is to rebuild indexes or to extend them incrementally. • AUTOSELECT • REBUILD • INCREMENTAL • DEFERRED

Table 21. DB2 UDB Load stage properties (continued)

Prompt	Type	Default	Description
Tracing Level	Long	0	Controls the type of tracing information that is added to the log. Use one or more of the following tracing levels: 0 - No Tracing 1 - Important events 2 - Performance 4 - Function You can combine the tracing levels, for example, a tracing level of 3 means that important events and performance messages are added to the log.
Copy loaded data	String List	None	Indicates that a copy of the data loaded into the server database is to be saved. Use this property when the server database is configured with LOGRETAIN or USEREXIT. Use one of the following values: **No** - No copy is made. **Yes** - A copy is made. **Use ADSM** - A copy is made using ADSTAR Distributed Storage Manager (ADSM). "Use ADSM" is available for IBM DB2 V6 servers configured with forward recovery. **Use TSM** - A copy is made using Tivoli® Storage Manager (TSM). "Use TSM" is available for IBM DB2 V7 servers configured with forward recovery. The copy options are available for both immediate and delayed load modes. • No • Yes • Use ADSM • Use TSM
Copy To device/directory name	String	None	The name of a device or path name where the copy is generated. This property is valid only when **Copy loaded data** is **Yes**.
Copy Load library name	String	None	The name of the shared library of the vendor product that is used to generate the copy. This property is valid only when **Copy loaded data** is **Yes**.

Table 21. DB2 UDB Load stage properties (continued)

Prompt	Type	Default	Description
Allow access mode	String List	NO	Specifies the locking level for the target table. **NO** - The load locks the target table for exclusive access during the load. **READ** - The load locks the target table in a share mode. This option is available at IBM DB2 Server version 8 only. (NO or READ)
Use table space for allow read access	String	None	A modifier that is supported by ALLOW READ ACCESS. The specified table space is used for building a shadow copy of the index if the indexes are being rebuilt. The shadow copy of the index is copied back to the original table space at the end of the load during an INDEX COPY PHASE. Only system temporary table spaces can be used with this option. This option is available at IBM DB2 Server version 8 only.
Check pending cascade	String List	DEFERRED	Specifies whether or not the check pending state of the loaded table is immediately cascaded to all descendents. **IMMEDIATE** - Indicates that the check pending state for foreign key constraints is immediately extended to all descendent foreign key tables. **DEFERRED** - Indicates that only the loaded table will be placed in the check pending state. This option is available at IBM DB2 Server version 8 only. • IMMEDIATE • DEFERRED
Lock with force	String List	NO	Allows load to force off other applications that hold conflicting locks. **NO** - The load waits, that is, does not force off other applications. **YES** - The load forces off other applications that hold conflicting locks to allow the load utility to proceed. This option requires the same authority as the FORCE APPLICATIONS database command. • NO • YES

Table 21. DB2 UDB Load stage properties (continued)

Prompt	Type	Default	Description
Partitioned DB Configuration	String List	NO	Indicates a load into a partitioned table. **NO** - The PARTITIONED DB CONFIG parameter is not in the LOAD command. **YES** - The PARTITIONED DB CONFIG parameter is in the LOAD command. This option is available at IBM DB2 Server version 8 only. • NO • YES
HOSTNAME	String	None	The host name for the file transfer command. If not specified, "nohost" is used. This option is available at IBM DB2 Server version 8 only.
FILE_TRANSFER_CMD	String	None	Specifies a file executable, batch file, or script that is called before data is loaded into any partitions. The value specified must be a fully qualified path. The full path, including the execution file name, must not exceed 254 characters. Refer to IBM DB2 documentation for additional information. This option is available at IBM DB2 Server version 8 only.
PART_FILE_LOCATION	String	None	The fully qualified directory where the partitioned files are located. Refer to IBM DB2 documentation for additional information. This option is available at IBM DB2 Server version 8 only.
OUTPUT_DBPARTNUMS	String	None	A list of partition numbers. The partition numbers represent the database partitions on which the load operation is performed. The partition numbers must be a subset of the database partitions on which the table is defined. The items in the list must be separated by commas. Ranges are permitted. Refer to IBM DB2 documentation for additional information. This option is available at IBM DB2 Server version 8 only.
PARTITIONING_ DBPARTNUMS	String	None	A list of partition numbers that are used in the partitioning process. Use commas to separate the items in the list. Ranges are allowed. If not specified, the LOAD command determines how many partitions are needed and which partitions to use in order to achieve optimal performance. Refer to IBM DB2 documentation for additional information. This option is available at IBM DB2 Server version 8 only.

Table 21. DB2 UDB Load stage properties (continued)

Prompt	Type	Default	Description
MODE	String List	PARTITION_ AND_LOAD	Specifies the load mode for partitioned databases. Refer to IBM DB2 documentation for additional information. This option is available at IBM DB2 Server version 8 only. • PARTITION_AND_LOAD • PARTION_ONLY • LOAD_ONLY • LOAD_ONLY_ VERIFY_PART • ANALYZE
MAX_NUM_ PART_AGENTS	Integer	25	The maximum number of partitioning agents to be used in a load session. This option is available at IBM DB2 Server version 8 only.
ISOLATE_PART_ ERRORS	String List	LOAD_ERRS_ ONLY	Indicates how the load operation reacts to errors that occur on individual partitions. Refer to IBM DB2 documentation for additional information. This option is available at IBM DB2 Server version 8 only. • SETUP_ERRS_ONLY • LOAD_ERRS_ONLY • SETUP_AND_LOAD_ERRS • NO_ ISOLATION
STATUS_ INTERVAL	Integer	100	Specifies the number of megabytes of data to load before generating a progress message. Valid values are whole number in the range of 1 to 4000. This option is available at IBM DB2 Server version 8 only.
PORT_RANGE	String	None	The range of TCP ports used to create sockets for internal communications. The default range is from 6000 to 6063. This option is available at IBM DB2 Server version 8 only.
CHECK_ TRUNCATION	String List	NO	Specifies whether the data records should be checked for truncation at input/output. **YES** - The program checks for truncation of data records at input/output. **NO** - The program does not check for truncation of data records at input/output. This option is available at IBM DB2Server version 8 only.
MAP_FILE_INPUT	String	None	The name of the input partitioning map file. The partitioning map file must be specified if the partitioning map is customized. This option is available at IBM DB2 Server version 8 only.
MAP_FILE_OUTPUT	String	None	The name of the output partitioning map file. The partitioning map file must be specified when **MODE** is set to **ANALYZE**. This option is available at IBM DB2 Server version 8 only.

Table 21. DB2 UDB Load stage properties (continued)

Prompt	Type	Default	Description
TRACE	Integer	0	Specifies the number of records to trace when you require a review of a dump of the data conversion process and the output of the hashing values. This option is available at IBM DB2 Server version 8 only.
NEWLINE	String List	NO	Specifies if each record is to be checked for a newline character. This option is used when the input data file is an ASC file with each record delimited by a newline character and **File type modifier** is set to **RECLEN = x**. **YES** - Each record is checked for a newline character. The record length is also checked. **NO** - Records are not checked for a newline character. This option is available at IBM DB2 Server version 8 only. • YES • NO
DISTFILE	String	None	The name of the partitioned distribution file. This option is available at IBM DB2 Server version 8 only.
OMIT_HEADER	String List	NO	Specifies whether a partition map header should be included in the partition file. **YES** - The OMIT_HEADER key word is in the LOAD command, and a partition map header should not be included in the partition file. **NO** - A partition map header is included in the partition file. This option is available at IBM DB2 Server version 8 only. • YES • NO
RUN_STAT_DBPARTNUM	Integer	-1	Specifies on which database partition to collect statistics. If **Statistics** is set to any value other than **NoStatistics**, statistics are collected only on one database partition. If **RUN_STAT_DBPARTNUM** is set to **-1**, statistics are collected on the first database partition in the output partition list. This option is available at IBM DB2 Server version 8 only.

* The property value is used by the *sqluload* API program which calls the *uload* utility.

Chapter 7. DB2Z stage

You can use the IBM DB2Z stage in your job design to read and load data.

About this task

When you use IBM InfoSphere DataStage to read and load DB2 data, you can choose from a collection of connectivity options. For most new jobs, use the DB2 Connector stage, which offers better functionality and performance than the DB2Z stage.

If you have jobs that use the DB2 DB2Z stage and want to use the connector, use the Connector Migration Tool to migrate jobs to use the connector.

Developing DB2Z stage jobs

You can use the IBM InfoSphere DataStage and QualityStage Designer client to define a DB2Z stage that accesses an IBM DB2 for z/OS database.

Procedure

1. "Importing metadata" on page 110 from a DB2 source.
2. "Accessing the DB2z stage from InfoSphere DataStage" on page 111 from the Designer client.
3. To set up the DB2Z stage to read data:
   a. "Configuring the DB2Z stage as a source" on page 112.
   b. "Setting up column definitions" on page 111.
   c. "Defining usage properties for reading data" on page 113
4. To set up the DB2Z stage to load data:
   a. "Defining connection properties for loading data" on page 113.
   b. "Setting up column definitions" on page 111 if they are not already specified for the link.
   c. "Defining target properties for loading data" on page 114.

Working with metadata

You can import metadata through a connector and save it so that the metadata is available to the local project and to other projects and components.

With the DB2z stage, you can import the following information:

- Data sources
- Database tables, system tables, and views for a particular data source (optionally, with fully qualified names or aliases)
- Descriptions of columns in a table

When you import metadata, information about the database columns is collected, including the names of columns, the length of columns, and the data types of columns.

Importing metadata

To place table definitions in the dynamic repository where they can be used by other projects or components, use the IBM InfoSphere DataStage and QualityStage Designer client to import metadata by using the DB2Z stage. When you import metadata from a DB2Z data source, a table is created in the dynamic repository, and a table definition is created in your project repository tree.

Procedure

1. From the Designer client, open the Import Connector Metadata wizard by selecting **Import** > **Table Definitions** > **Start Connector Import Wizard** from the main menu.
2. On the Data Source Location page, select the host name and database that identifies where you want to store the metadata in the dynamic repository, and click **Next**. If the lists are not populated, click **New location** to start the Shared Metadata Management tool.
3. On the Connector Selection page, select the DB2Z stage for the import process, and click **Next**.
4. On the Connection Details page, specify the connection details for the data source, and click **Next**. The subsequent pages collect information that is specific to the type of connector that you are using for the import process.
5. Specify the details for the DB2Z stage that you selected.
6. Confirm the import details, and click **Import**.
7. Browse the repository tree, and select the location in the project repository for the table definition that you are creating. Click **OK**.

Saving user-defined metadata

You can define and save metadata for columns by using the stage editor to modify the column information for a link. After you save the metadata, you can load the column definitions from the repository.

Procedure

1. On IBM InfoSphere DataStage parallel canvas, double-click the **DB2Z stage** icon.
2. In the top, left corner of the stage editor, click the link that you want to define and save metadata for.
3. On the **Columns** tab, define metadata for the columns.
4. To save the column definitions as a table definition in the repository, click **Save**.
5. Enter the appropriate information in the Save Table Definition window, and click **OK**.
6. In the Save Table Definition As window, select the folder where you want to save the table definition, and click **Save**.

Modifying stage and link attributes

When you include a DB2Z stage in an IBM InfoSphere DataStage job, the stage and link automatically obtain basic, default attributes. You can modify some of the default attributes and specify some other basic information.

Procedure

1. To specify or modify the attributes of the DB2Z stage and link, double-click the stage icon in InfoSphere DataStage parallel canvas to open the stage editor.

2. You can enter and modify the following attributes:

   Stage name, Input name or Output name
   Modify the default name of the stage or the link. Alternatively, you can modify the name of the stage or link in InfoSphere DataStage parallel canvas.

   Description
   Enter an optional description of the stage or link.

Accessing the DB2z stage from InfoSphere DataStage

Use the IBM InfoSphere DataStage and QualityStage Designer client to access the DB2z stage by placing the icon for the stage on the parallel canvas.

Procedure

1. From the Designer client, select **File** > **New** from the menu.
2. In the New window, select the **Parallel Job** icon, and click **OK**.
3. On the left side of the Designer client in the Palette menu, select the **Database** category.
4. Drag the DB2Z stage icon to the parallel canvas.

Configuring data source connections

To perform read or write operations by using the DB2Z stage, you must define a connection to an IBM DB2 for z/OS data source.

Procedure

1. On InfoSphere DataStage parallel canvas, double-click the **DB2Z stage** icon.
2. To display the properties for the job that you are designing, click the **Input** or **Output** tab on the stage editor, as appropriate.
3. On the **Properties** tab in the **Connection** section, specify the instance, database, user name, and password that you want to use to make the connection.
4. In the **Database Name** field, specify the name of the DB2 for z/OS database.
5. In the **Database Password**, specify the password for the DB2 for z/OS database.
6. In the **Database User ID**, specify the user ID for the DB2 for z/OS database.
7. Click **OK**.

Setting up column definitions

You set up column definitions for read operations and load operations in a similar way. You can also customize the columns grid, save column definitions for later use, and load predefined column definitions from the repository.

Procedure

1. On IBM InfoSphere DataStage parallel canvas, double-click the **DB2Z stage** icon.
2. In the stage editor, click the **Input** or **Output** tab.
3. On the **Columns** tab, modify the columns grid to specify the metadata that you want to define.
   a. Right-click within the grid, and select **Properties** from the menu.
   b. In the Grid properties window, select the properties that you want to display and in what order. Then, click **OK**.

4. Enter column definitions for the table by using one of the following methods:

Option	Description
Method 1	1. In the **Column name** column, double-click inside the appropriate cell and type a column name. 2. For each cell in the row, double-click inside the cell and select the options that you want. 3. In the **Description** column, double-click inside the appropriate cell and type a description.
Method 2	1. Right-click within the grid, and select **Edit row** from the menu. 2. In the Edit column metadata window, enter the column metadata.

   Choose Method 1 if you want to enter column definitions. If you choose Method 2, you can enter column metadata.
5. To share metadata between several columns, select the columns that you want to share metadata.
   a. Right-click and select **Propagate values**.
   b. In the Propagate column values window, select the properties that you want the selected columns to share.
6. To save the column definitions as a table definition in the repository, click **Save**.
   a. Enter the appropriate information in the Save Table Definition window, and then click **OK**.
   b. In the Save Table Definition As window, select the folder where you want to save the table definition, and then click **Save**.
7. To load column definitions from the repository, click **Load**.
   a. In the Table Definitions window, select the table definition that you want to load, and then click **OK**.
   b. In the Select Columns window, use the arrow buttons to move columns from the **Available columns** list to the **Selected columns** list. Then, click **OK**.

Reading data

You can use IBM DB2Z stage to read data from a DB2 for z/OS table.

Before you can read data from a DB2 for z/OS table, configure the DB2Z stage to process data as a source.

Configuring the DB2Z stage as a source

By configuring the IBM DB2Z stage to process data as a source, you can use the DB2Z stage to read data.

Procedure

1. On InfoSphere DataStage parallel canvas, double-click the **DB2Z stage** icon.
2. In the top left corner of the stage editor, select the output link that you want to edit.

3. On the **Properties** tab in the **Connection** section, configure the data source connection for the DB2Z stage.
4. Choose one of the following options to load, save, or test the data connection:

Option	Description
Load	Loads an existing data connection from the repository.
Save	Saves the connection settings that you specified.

Defining usage properties for reading data

Define the usage properties for a read operation to use the IBM DB2Z stage as a source for reading data.

Before you begin

You must configure a database connection for the DB2Z stage.

Procedure

1. On InfoSphere DataStage parallel canvas, double-click the **DB2Z stage** icon.
2. In the top left corner of the stage editor, select the output link that you want to edit.
3. On the **Properties** tab in the **Source** section, specify how the stage operates in a job. Specify whether you want to read using a **Table Name** or **User defined SQL**.
4. In the **Table Name** field, specify the table that you want to read.
5. In the **SQL Query** field, specify the SQLstatement to read data.

Loading data

You can use the DB2Z stage to load data into an IBM DB2 for z/OS table.

To load data into a DB2 for z/OS table by using the DB2Z stage, configure the DB2Z stage to process data as a target.

Defining connection properties for loading data

By configuring the DB2Z stage to process data as a target, you can use the DB2Z stage to load data.

Procedure

1. On the parallel canvas, double-click the **DB2Z stage** icon.
   a. The link that you create has a system-generated name. You can edit the link name either in the IBM InfoSphere DataStage parallel canvas or in the stage editor.
2. In the top left corner of the stage editor, select the output link that you want to edit.
3. On the **Properties** tab.
4. In the **Connection** section, specify the following connection settings for the DB2Z stage:
   •

Database Name
Enter the database name for the DB2 for z/OS database. Use the same name that is cataloged in the local DB2 system or in IBM DB2 Connect™.

Database user ID
Enter the user ID for connecting to the DB2 for z/OS system.

Database password
Enter the password for connecting to the DB2 for z/OS system.

Defining target properties for loading data

Define the usage properties for a load operation to use the IBM DB2Z stage as a target for loading data.

Before you begin

You must configure a database connection for the DB2Z stage.

Procedure

1. On InfoSphere DataStage parallel canvas, double-click the **DB2Z stage** icon.
2. In the top left corner of the stage editor, select the input link that you want to edit.
3. Click the **Properties** tab.
4. Specify the **Mode** in the following choices:
   - **Append:** New rows get added during the load operation.
   - **Truncate** Existing row gets deleted.
5. In the **Table Name** field, specify the name of the destination table that is used in the SQL statements that are meant for writing data.

Defining transfer properties for loading data

Define the transfer properties for a load operation to transfer data.

Procedure

1. On IBM InfoSphere DataStage parallel canvas, double-click the **DB2Z stage** icon.
2. In the top left corner of the stage editor, select the input link that you want to edit.
3. Click the **Properties** tab.
4. Specify the following properties in order to transfer data:
   -

Overwrite
When this property is set to **True**, any existing MVS datasets get overwritten. If this property is set to **False**, the job gets terminated. The default option is **False**

Transfer To
Specify the name of the computer you want to transfer data. This is a mandatory property.

Transfer Type
Specify the type of data transfer. The only option available for data transfer is **FTP**.

User Specify the user name for transferring data.

Password
Specify the password for the user name to transfer data.

Character Set
Determines the EBCDIC character set for NCHAR data to be transferred to z/OS.

Defining options properties for loading data

Define the options properties for a load operation.

Before you begin

You must configure a database connection for the DB2Z stage.

Procedure

To define the options properties for a load operation, perform the following:

1. On IBM InfoSphere DataStage parallel canvas, double-click the **DB2Z stage** icon.
2. In the top left corner of the stage editor, select the input link that you want to edit.
3. Click the **Properties** tab.
4. Specify the properties for options category.

Options

Specify the properties for the IBM® DB2Z stage to perform bulk load jobs. Set options to create files IBM® DB2Z stage on z/OS, to load a particular partition, and to display statistics at the end of the load.

Select the properties for the data file attributes for the IBM DB2Z stage. Select **Yes** to add properties from the list of properties.

Batch pipe system ID
Specify the name of the Batch pipe system ID.

This property determines how the data is transferred to z/OS. The Batch pipe system ID property is optional. If you specify the ID, then Batch pipes are used to transfer the data to z/OS.

Close command
Use the Close command property to execute an SQL statement after the LOAD utility completes successfully. The Close command property is optional.

Close command error action
Use the Close command error action property to identify the action to take if the Close command fails.

The Close command error action property is optional. Valid values are Fail and Continue. Fail is the default value.

Coded character set identifier(s)
Specifies up to three coded character set identifiers (CCSIDs) for the input file. The first value specifies the CCSID for SBCS data that is found in the input file, the second value specifies the CCSID for mixed DBCS data, and the third value specifies the CCSID for DBCS data.

DSN prefix

An MVS data set name prefix is used to construct names of data sets or Batch pipes. If you do not specify a name, the database user ID is used as a prefix and a suitable name is constructed.

- Data files are named as *prefix.IN#####*, where ##### is the partition number. If the table being loaded is not partitioned, then use 00000 for the partition number.
- Discard files are named as *prefix.DSC#####*, where ##### is the partition number. If the table being loaded is not partitioned, then use 00000 for the partition number.
- Work files are named as *prefix.WORK1* and *prefix.WORK2*.
- Error files are named as *prefix.SYSERR*.
- Map files are named as *prefix.SYSMAP*.

Files only

Use the Files only property to create data files on z/OS.

The Files only property is optional. Valid values for this property are Yes and No. If you select Yes, data files are created on z/OS but the load utility is not invoked.

Note: When a BatchPipes® subsystem ID is specified, you cannot select Yes.

Image-copy function

Use the properties in this group to specify details about image copy and recovery files. Specify whether to run an Image-copy function after completing a bulk load job.

The valid values for this property are:

- Concurrent
- Full
- Incremental
- No

No is the default value. You can configure the Image-copy function by specifying values to the properties that are displayed when you select a valid value.

Load with logging

Use the Load with logging property to indicate whether logging must occur during the load process.

The Load with logging property is optional. The valid values are No and Yes. No is the default value.

Modify discard dataset properties

Specify the discard data set properties for the load utility. To modify the discard data set properties, select Yes. For more information, see "Data set properties" on page 118.

Modify error dataset properties

Specify the error data set properties for the load utility. To modify the error data set properties, select Yes. For more information, see "Data set properties" on page 118.

Modify map dataset properties
Specify the Map data set properties for the load utility. To modify the map data set properties, select Yes. For more information, see “Data set properties” on page 118.

Modify work1 dataset properties
Specify the work1 data set properties for the load utility. This is a temporary data set. To specify the map data set properties, select Yes. For more information, see “Data set properties” on page 118.

Modify work2 dataset properties
Specify the work2 data set properties for the load utility. This is a temporary data set. To modify the map data set properties, select Yes. For more information, see “Data set properties” on page 118.

Open command
Use the Open command property to execute an SQL statement before the load utility begins.

The Open command property is optional.

Open command error action
Use the Open command error action property to identify the action to take if the Open command fails.

The Open command error action property is optional. Valid values for this property are Fail and Continue. Fail is the default value.

Partition number
Specify the partition to be loaded.

The Partition number property is optional. The value for this property must be an integer. If you do not specify a value, the data is loaded into all partitions.

Row count estimate
Use the Row count estimate property to provide the estimated number of rows to be loaded into all the partitions combined.

This estimate is used to calculate the amount of disk space to allocate for the data sets. The Row count estimate is optional. The value for this property must be an integer. The default value is 1000.

Set copy-pending
Use the Set copy-pending property to specify whether the table space is set to copy-pending status.

The Set copy-pending property is optional. Valid values for this property are No and Yes. The default value is No.

Note: Use this property when the value for Load with logging is No.

Statistics
Specify whether to display statistics for the bulk load at the end of the load job. This property is optional.

The default value is None.

Utility ID
The Utility ID is a unique identifier that is within DB2 to identify the execution of the load utility.

The Utility ID property is optional and the default value is DB2ZLOAD.

Verbose
Specify whether to display information and processing messages during the bulk load.

Data set properties:

Assign the attributes to the data set properties to load data in to DB2 on IBM z/OS.

Abnormal termination
Specify the action to take with the data set at abnormal job termination.

Data class
Specify the SMS data class (DATACLAS).

This property is optional. The value for this property must be a string value.

Image copy backup file
Specify whether to create an image-copy file. The Image-copy Backup File property is optional. The value for this property should be boolean.

Dataset name
Specify a name for the Image copy data set. This property is optional. The value for this property must be a string value.

Image copy recovery file
Specify whether to create the image-copy recovery file.

Management class
Specify the SMS management class (MGMTCLAS).

This property is optional. The value for this property must be a string value.

Normal Termination
Specify the action to take with the data set at normal job termination.

Number of buffers
Specify the number of buffers. This property is optional. The value for this property must be an integer.

Primary allocation
Specify the z/OS disk space primary allocation amount. The range of values is from 1 to 1677215.

Secondary allocation
Specify the z/OS disk space secondary allocation amount. The range of values is from 1 to 1677215.

Space type
Specify the z/OS disk space allocation type.

The Space type property is optional. Valid values are Cylinders and Tracks. The default value is Cylinders.

Status Specify the disposition status of input data set used by LOAD utility. This property is disabled when Batch pipe system ID has any value. The valid values for this property are:

- Replace: Deletes an existing data set and creates a new data set.
- New: Indicates that the file does not currently exist.
- Old: Overwrites an existing data set or fails if the data set does not exist.

- Share: Identical to Old except that several jobs may read from the data set at the same time
- Append: Appends to the end of an existing data set or creates a new data set if it does not already exist.

The default value is Replace.

Storage class
Specify the SMS storage class (STORCLAS).

This property is optional. The value for this property must be a string value.

Unit Specify the device number, device type, or group name for the data set.

Volume(s)
Specifies the list of volume serial numbers for this allocation. The values can be entered with or without enclosing them in parenthesis.

This property is optional. The value for this property must be a string value or a list of string values that are separated by commas.

Compiling and running a DB2Z stage job

You compile jobs containing a IBM DB2Z stage into executable scripts that you schedule by using the InfoSphere DataStage and QualityStage Director client and run on InfoSphere DataStage.

Procedure

1. In the InfoSphere DataStage and QualityStage Designer client, open the job that you want to compile.
2. Click the **Compile** toolbar button.
3. If the Compilation Status area shows errors, edit the job to resolve the errors. After resolving the errors, click the **Re-compile** button.
4. When the job compiles successfully, click the **Run** toolbar button, and specify the job run options:
   a. Enter the job parameters as required.
   b. Click the **Validate** button to verify that the job will run successfully without actually extracting, converting, or writing data.
   c. Click the **Run** button to extract, convert, or write data.
5. To view the results of validating or running a job:
   a. In the Designer client, select **Tools** > **Run Director** to open the Director client.
   b. In the Status column, verify that the job was validated or completed successfully.
   c. If the job or validation fails, select **View** > **Log** to identify any runtime problems.
6. If the job has runtime problems, fix the problems, recompile, validate (optional), and run the job until it completes successfully.

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 22. SQL statement types

SQL statement	Description
SELECT	Selects rows of data from a database table. The query can perform joins between multiple tables and aggregations of values in columns.
INSERT	Inserts rows in a database table.
UPDATE	Updates existing rows in a database table.
DELETE	Deletes rows from a database table.

You can use the SQL builder 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.

Avoid using column names that are SQL reserved words as their use might result in unexpected results when the SQL is built or run.

Starting SQL builder from a stage editor

If a stage supports the SQL builder, you can open the SQL builder by clicking **Build SQL** in the stage editor. For some stages, you can use the SQL builder only for some access methods.

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 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 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 upper half of the SQL builder are common to all types of SQL statement that you can build. The pages that are available in the lower half depend on the type of query that you build.

Toolbar

The toolbar for the SQL builder contains tools for actions such as clearing the current query, viewing data, and validating the statement.

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

The tree panel shows the table definitions in the IBM InfoSphere DataStage repository. You can import a table definition from the database that you want to query.

You can import the table definition by using 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

The table selection canvas shows a list of columns and column types for the table that the SQL statement accesses.

You can 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`.

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

Use the Selection page to specify details for a SELECT statement.

Column selection grid

Use the column selection grid to specify the columns to include 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 list in the grid
- Double-click the column name in the table selection canvas
- Copy and paste from the table selection canvas

Column expression

The column expression identifies the columns to include in the SELECT statement.

You can specify the following parts:

- **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

This property 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 list.

To specify the table name at run time, choose a job parameter from the list.

Column alias

Use this property to specify an alias for the column.

Output

Select this property to indicate that the column is part of the query output. The property is selected automatically 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 adds an ORDER BY clause to the query.

Sort order

You can specify the order in which rows are sorted if you order by more than one column.

Shortcut menu

Use the shortcut menu to paste a column that you copied from the table selection canvas, insert or remove a row, and show or hide the filter panel.

Filter panel

In the filter panel, you specify a WHERE clause for the SELECT statement that you are building. The filter panel includes a predicate list and an expression editor panel, the contents of which depends on the chosen predicate.

Filter expression panel

The filter expression panel shows the filters that you added to the query. You can edit a filter that you added by using the filter expression editor or you can enter a filter manually.

Group page

Use the Group page, which appears when you build SELECT statements, to specify that the results of the query are grouped by a column or columns.

Also, you can use the page to aggregate the results in some of the columns. For example, you can specify COUNT to count the number of rows that contain a non-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

In the grouping grid, you can specify the columns to group by or aggregate 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 that are available depend on the stage that you opened the SQL builder from. All SQL syntax variants include the AVG, COUNT, MAX, MIN, STDDEV, and VARIANCE aggregation functions.

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

In the Filter panel, you can specify a HAVING clause for the SELECT statement. The Filter panel includes a predicate list and an expression editor panel, the contents of which depends on the chosen predicate.

Filter Expression panel

The Filter Expression panel shows the filters that you added to the query. You can edit a filter that you added by using the filter expression editor, or you can enter a filter manually.

Insert page

Use the Insert page to specify the details of an INSERT statement. The page includes the insert columns grid.

Insert Columns grid

In the Insert Columns grid, you specify the columns to include in the INSERT statement and the values that they will take.

Insert column

This property identifies the columns to include in the INSERT statement.

You can populate this in a number of ways:

- Drag columns from the table in the table selection canvas
- Choose columns from a list in the grid
- Double-click the column name in the table selection canvas
- Copy and paste from the table selection canvas

Insert value

This property identifies the values that you are setting the corresponding column to. You can enter a value manually or specify a job parameter, expression, data flow variable, or lookup column.

When you specify a value, you can use the following objects:

- **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

Use the Update page to specify details of an UPDATE statement.

Update Column grid

In the Update Column grid, you specify the columns to include in the UPDATE statement and the values that they will take.

Update column

This property identifies the columns to include in the UPDATE statement.

You can populate this in the following ways:

- Drag columns from the table in the table selection canvas.
- Choose columns from a list in the grid.
- Double-click the column name in the table selection canvas.
- Copy and paste from the table selection canvas.

Update value

This property identifies the value that you are setting the corresponding column to. You can enter a value in the field manually, or you can specify a job parameter, expression, data flow variable, or lookup column.

You can specify the following objects:

- **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

In the filter panel, you can specify a WHERE clause for the UPDATE statement that you build. The filter panel includes a predicate list and an expression editor panel, the contents of which depends on the chosen predicate.

Filter expression panel

The filter expression panel shows the filters that you added for the query. You can edit the filter in the panel or enter a filter manually.

Delete page

On the Delete page, you specify details for the DELETE statement that you build.

Filter panel

On the filter panel, you can specify a WHERE clause for the DELETE statement that you build. The filter panel includes a predicate list and an expression editor panel, the contents of which depend on the chosen predicate.

Filter expression panel

The filter expression panel shows the filters that you add to the query. You can edit a filter in the panel or enter a filter manually.

SQL page

On the SQL page, you view the SQL statement that you build.

For SELECT queries, if the columns that you 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 that you loaded in a stage do not match the columns that are generated by the SQL statement that you built, you can reconcile the differences in the Resolve columns grid.

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

In the expression editor, you can specify a WHERE clause to add to your SQL statement. If you are joining tables, you can also specify a WHERE or HAVING clause for a join condition.

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

In the expression editor, you can specify a filter that uses the Between, Comparison, In, Like, or Null predicates.

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 predicate

When you specify a Between predicate in the expression editor, you choose a column, specify a range, and specify whether a value must be in the range or not in the range.

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 predicate

When you specify a Comparison predicate in the expression editor, you choose a column, a comparison operator, and a comparison value.

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 predicate

When you specify an In predicate in the expression editor, you choose a column, select items to include in the query, and specify whether selected values are in the list or not in the list.

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 predicate

When you specify a Like predicate in the expression editor, you choose a column, an operator, and a value. You then specify whether values are included or excluded by the comparison.

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 predicate

When you specify a Null predicate in the expression editor, you choose a column and specify whether your query must match a NULL or NOT NULL condition in the column.

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 predicate

When you specify a Join predicate in the expression editor, you choose the columns to join and a join type.

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, and case expression editor

In this version of the expression editor, you can specify an expression in a WHERE expression, a HAVING expression, or a join condition. The expression editor windows are numbered to show how deeply they are nested.

Calculation predicate

When you use the Calculation predicate, you specify the left value, right value, and calculation operator in the expression editor.

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 predicate

When you use the functions predicate, you can specify the function, description, and function parameters in the expression editor.

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 predicate

When you use the case predicate, you can include case statements in the SQL that you build in the expression editor.

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

From the expression editor, you can open a menu where you can specify details about an argument in the expression.

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.

  Selecting this causes the Function dialog box to open.

- **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 window

In the Functions Form window, you select a function to use in 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 window:

In the Function window, you can select a function to use in 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 window

This window lists the job parameters that are currently defined for the job and the data type of each parameter. The SQL builder does not check that the type of parameter that you insert matches the type that is expected by the argument that you use it for.

Joining tables

When you use the SQL builder to build SELECT statements, you can specify table joins in a statement.

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

When you add more than one table to the table selection canvas, the SQL builder inserts a join automatically. To change the join, you can use the Join Properties window, use the Alternate Relation window, or drag a column from one table to a column in another table.

You can change the join in the following ways:

- 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 window

Use the Join Properties window to change the type of an existing join and modify or specify the join condition.

The window 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 window

The Alternate Relation window shows the foreign key relationships that are defined between the target table and tables that appear to the left of it on the table selection canvas. Select the relationship that you want to appear as the join in your query so that it appears in the list box, and then click **OK**.

Properties windows

The Properties windows contain properties for tables, SQL, and joins.

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.

Table Properties window

In the Table Properties window, you can view the table name and view or edit the table alias.

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. 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 window

The SQL Properties window shows the SQL grammar that the SQL builder uses.

The SQL Properties window 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: DB2 connector

The DB2 Connector stage uses these environment variables.

CC_DB2_FILETYPEMODIFIERS

Set this environment variable to a string to specify the file type modifiers for the load process.

When the **Write mode** property is set to **Bulk load** and the **Bulk load with LOB or XML column(s)** property is set to **No**, set this environment variable to a string specifying the file type modifiers for the load process.

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_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_STRING_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. This environment variable applies to fields of Char, VarChar, and LongVarChar InfoSphere DataStage types.

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.

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. Because the information center uses XHTML, you can set display preferences in your browser. This 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:

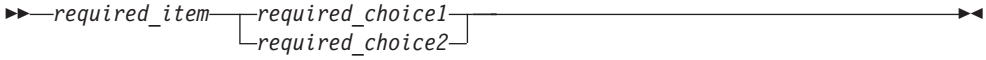
- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following conventions are used:
  - The >>--- symbol indicates the beginning of a syntax diagram.
  - The ---> symbol indicates that the syntax diagram is continued on the next line.
  - The >--- symbol 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).

- Optional items appear below the main path.

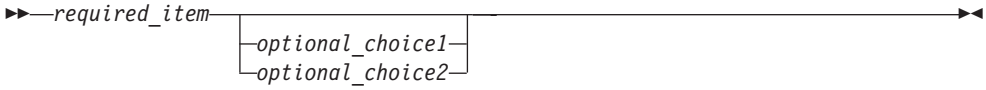
  If an optional item appears above the main path, that item has no effect on the execution of the syntax element and is used only for readability.

- If you can choose from two or more items, they appear vertically, in a stack.

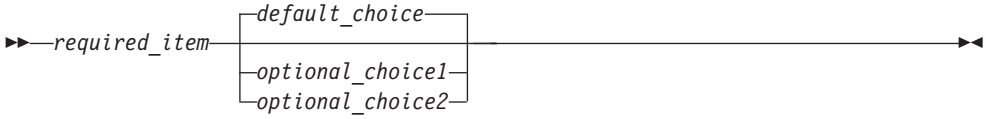
  If you must choose one of the items, one item of the stack appears on the main path.



  If choosing one of the items is optional, the entire stack appears below the main path.

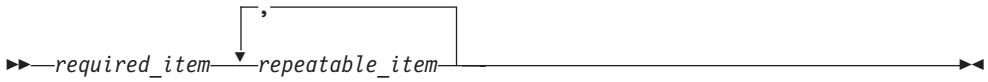


  If one of the items is the default, it appears above the main path, and the remaining choices are shown below.



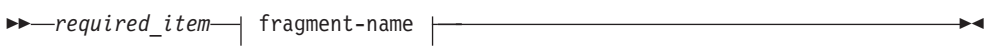
- An arrow returning to the left, above the main line, indicates an item that can be repeated.

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 23. IBM resources

Resource	Description and location
IBM Support Portal	You can customize support information by choosing the products and the topics that interest you at www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server
Software services	You can find information about software, IT, and business consulting services, on the solutions site at www.ibm.com/businesssolutions/
My IBM	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 www.ibm.com/account/
Training and certification	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 http://www.ibm.com/training
IBM representatives	You can contact an IBM representative to learn about solutions at www.ibm.com/connect/ibm/us/en/

Appendix E. Accessing the product documentation

Documentation is provided in a variety of formats: in the online IBM Knowledge Center, in an optional locally installed information center, and as PDF books. You can access the online or locally installed help directly from the product client interfaces.

IBM Knowledge Center is the best place to find the most up-to-date information for InfoSphere Information Server. IBM Knowledge 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 IBM Knowledge Center from the installed product or from a web browser.

Accessing IBM Knowledge Center

There are various ways to access the online documentation:

- Click the **Help** link in the upper right of the client interface.
- 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.
- Type the address in a web browser, for example, when you are not logged in to the product.

  Enter the following address to access all versions of InfoSphere Information Server documentation:

  `http://www.ibm.com/support/knowledgecenter/SSZJPZ/`

  If you want to access a particular topic, specify the version number with the product identifier, the documentation plug-in name, and the topic path in the URL. For example, the URL for the 11.3 version of this topic is as follows. (The ⇒ symbol indicates a line continuation):

  ```
  http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/⇒
  com.ibm.swg.im.iis.common.doc/common/accessingiidoc.html
  ```

  Tip:

  The knowledge center has a short URL as well:

  `http://ibm.biz/knowctr`

  To specify a short URL to a specific product page, version, or topic, use a hash character (#) between the short URL and the product identifier. For example, the short URL to all the InfoSphere Information Server documentation is the following URL:

  `http://ibm.biz/knowctr#SSZJPZ/`

  And, the short URL to the topic above to create a slightly shorter URL is the following URL (The ⇒ symbol indicates a line continuation):

  ```
  http://ibm.biz/knowctr#SSZJPZ_11.3.0/com.ibm.swg.im.iis.common.doc/⇒
  common/accessingiidoc.html
  ```

Changing help links to refer to locally installed documentation

IBM Knowledge Center contains the most up-to-date version of the documentation. However, you can install a local version of the documentation as an information center and configure your help links to point to it. A local information center is useful if your enterprise does not provide access to the internet.

Use the installation instructions that come with the information center installation package to install it on the computer of your choice. After you install and start the information center, you can use the **iisAdmin** command on the services tier computer to change the documentation location that the product F1 and help links refer to. (The ⇒ symbol indicates a line continuation):

Windows

```
IS_install_path\ASBServer\bin\iisAdmin.bat -set -key ⇒
com.ibm.iis.infocenter.url -value http://<host>:<port>/help/topic/
```

AIX Linux

```
IS_install_path/ASBServer/bin/iisAdmin.sh -set -key ⇒
com.ibm.iis.infocenter.url -value http://<host>:<port>/help/topic/
```

Where <host> is the name of the computer where the information center is installed and <port> is the port number for the information center. The default port number is 8888. For example, on a computer named server1.example.com that uses the default port, the URL value would be http://server1.example.com:8888/help/topic/.

Obtaining PDF and hardcopy documentation

- The 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.

Appendix F. Providing feedback on the product documentation

You can provide helpful feedback regarding IBM documentation.

Your feedback helps IBM to provide quality information. You can use any of the following methods to provide comments:

- To provide a comment about a topic in IBM Knowledge Center that is hosted on the IBM website, sign in and add a comment by clicking **Add Comment** button at the bottom of the topic. Comments submitted this way are viewable by the public.
- To send a comment about the topic in IBM Knowledge Center to IBM that is not viewable by anyone else, sign in and click the **Feedback** link at the bottom of IBM Knowledge Center.
- Send your comments by using the online readers' comment form at 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. This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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. 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.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session or persistent cookies. If a product or component is not listed, that product or component does not use cookies.

Table 24. Use of cookies by InfoSphere Information Server products and components

Product module	Component or feature	Type of cookie that is used	Collect this data	Purpose of data	Disabling the cookies
Any (part of InfoSphere Information Server installation)	InfoSphere Information Server web console	• Session • Persistent	User name	• Session management • Authentication	Cannot be disabled
Any (part of InfoSphere Information Server installation)	InfoSphere Metadata Asset Manager	• Session • Persistent	No personally identifiable information	• Session management • Authentication • Enhanced user usability • Single sign-on configuration	Cannot be disabled

Table 24. Use of cookies by InfoSphere Information Server products and components (continued)

Product module	Component or feature	Type of cookie that is used	Collect this data	Purpose of data	Disabling the cookies
InfoSphere DataStage	Big Data File stage	• Session • Persistent	• User name • Digital signature • Session ID	• Session management • Authentication • Single sign-on configuration	Cannot be disabled
InfoSphere DataStage	XML stage	Session	Internal identifiers	• Session management • Authentication	Cannot be disabled
InfoSphere DataStage	IBM InfoSphere DataStage and QualityStage Operations Console	Session	No personally identifiable information	• Session management • Authentication	Cannot be disabled
InfoSphere Data Click	InfoSphere Information Server web console	• Session • Persistent	User name	• Session management • Authentication	Cannot be disabled
InfoSphere Data Quality Console		Session	No personally identifiable information	• Session management • Authentication • Single sign-on configuration	Cannot be disabled
InfoSphere QualityStage Standardization Rules Designer	InfoSphere Information Server web console	• Session • Persistent	User name	• Session management • Authentication	Cannot be disabled
InfoSphere Information Governance Catalog		• Session • Persistent	• User name • Internal identifiers • State of the tree	• Session management • Authentication • Single sign-on configuration	Cannot be disabled
InfoSphere Information Analyzer	Data Rules stage in the InfoSphere DataStage and QualityStage Designer client	Session	Session ID	Session management	Cannot be disabled

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

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.

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^{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^{Link} licensee of the United States Postal Service.

Other company, product or service names may be trademarks or service marks of others.

Index

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Printed in USA

SC19-4256-00

Spine information:

IBM InfoSphere DataStage and QualityStage **Version 11 Release 3** **Connectivity Guide for DB2 Databases** IBM