Data integration with Teradata using IBM InfoSphere Information Server

Skill Level: Intermediate

Jeff Li (liji@us.ibm.com)
Software Architect
IBM Corporation

Ravi Tatapudi (rtatapud@in.ibm.com)
Advisory Software Engineer
IBM

30 Nov 2009

This article introduces and describes the various Teradata integration solutions that are part of IBM® InfoSphere™ Information Server. It provides examples that demonstrate business integration scenarios and can be used as guides for solving typical Teradata integration issues.

Introduction

Teradata is one of the leading database systems for building enterprise data warehousing and analytical applications. It utilizes a massively parallel processing architecture and provides scalable solutions.

IBM InfoSphere Information Server is a unified and comprehensive information integration platform. It profiles, cleanses, and transforms data from heterogeneous data sources to deliver consistent and accurate business data. IBM Information Server is an ideal solution to integrate and synchronize Teradata enterprise data with other ERP systems and enterprise applications.

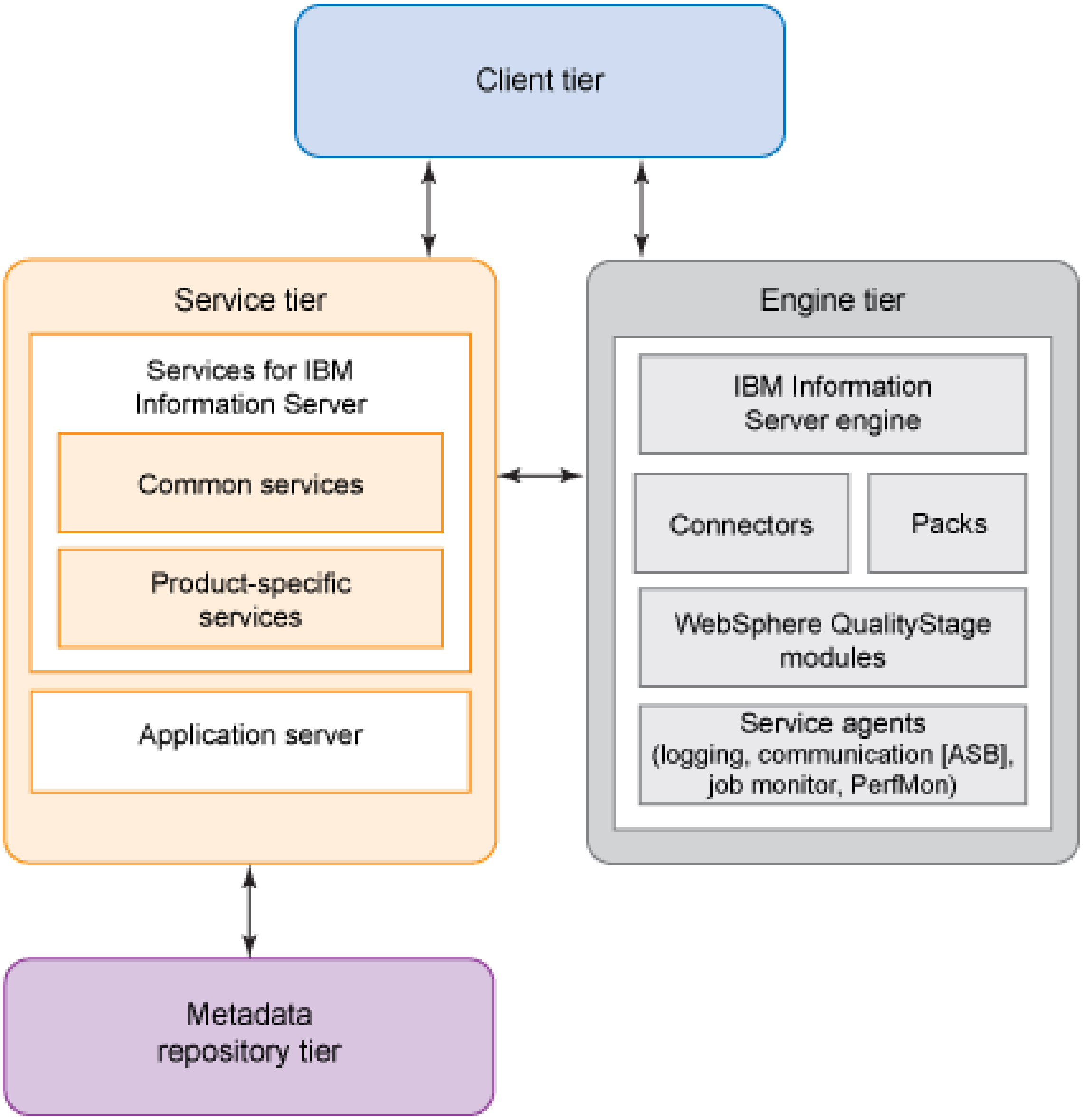
This article illustrates and compares various Teradata integration solutions within the IBM InfoSphere Information Server. It also contains examples to demonstrate the business integration scenarios. These examples provide guides that show you how to solve typical Teradata integration problems.

Product prerequisites and overview

As shown in Figure 1, The IBM InfoSphere Information Server can be viewed as having four logical tiers:

- Client
- Engine
- Metadata repository
- Services

Figure 1. IBM InfoSphere Information Server logical tiers


Client tier
Service tier
Services for IBM Information Server
Common services
Product-specific services
Application server
Engine tier
IBM Information Server engine
Connectors
Packs
WebSphere QualityStage modules
Service agents (logging, communication [ASB], job monitor, PerfMon)
Metadata repository tier

Each tier defines a logical group of software modules that can be mapped to a physical piece of hardware. The tiers can be installed separately on different computers or on the same computer. The Information Server supports deploying the engine tier on a symmetric multiprocessing (SMP) computing platform or on a massive parallel processing (MPP) computing platform to achieve high scalability and performance.

The InfoSphere Information Server requires the following components to support data integration with Teradata databases:

- DataStage Connectivity for Teradata — this component includes the Teradata Connector and all the other Teradata legacy stages. The

Teradata connector is a single solution designed to replace all the legacy stages. The Teradata connector is installed on the engine tier. The Teradata legacy stages, however, include the client installation for the client tier and the server installation for the engine tier.

- Teradata Tools and Utilities (TTU) — this component includes many products that work with the Teradata databases. TTU is required on the engine tier and must also be installed on the client tier if the Teradata legacy stages are used in ETL jobs. TTU depends on the following packages:
    - Teradata Generic Security Services (TeraGSS) client package
    - Teradata Shared Component for Internationalization (tdicu)
    - Teradata Call-Level Interface (CLIv2)
    - Teradata Named Pipes Access Module
    - Teradata FastExport Utility
    - Teradata FastLoad Utility
    - Teradata MultiLoad Utility
    - Teradata TPump Utility
    - Teradata Parallel Transporter Interface

The Teradata Connector leverages new Teradata features. It operates in either immediate access mode or in bulk mode:

- Immediate access mode — in this mode the connector sends the database SQL statements to the Teradata DBC/SQL partition and gets immediate responses back from the Teradata. The Teradata DBC/SQL partition is responsible for processing the SQL requests. The immediate mode and the DBC/SQL partition are suitable for supporting the low volume data processing.
- Bulk mode — this mode is suitable for batch and bulk data processing. In bulk mode, the connector leverages the Teradata parallel transporter interface and the multiple computing nodes defined in the SMP and MPP configuration to perform the parallel data load and data export operations. The Teradata parallel transporter interface is a parallel enabled programming interface that can run a bulk operation using multiple processes on multiple computing platforms. The Teradata connector supports four drivers specified in the Teradata parallel transporter interface:
    - Load driver — uses the FastLoad protocol to perform parallel load to

empty tables.

- Update driver — uses the Multiload protocol and supports parallel insert/update/delete/upsert operations to new or existing tables.
- Stream driver — uses the TPump protocol to perform the parallel real-time DML operations on tables. The stream driver uses row level locks. It allows applications to perform constant data load operations to a table in the background while the interactive read and write operations can occur concurrently.
- Export driver — uses the FastExport protocol to perform the parallel data export.

The Teradata Connector is available on IBM InfoSphere Information Server Version 8.0.1 and later. It is designed as a single solution to replace all the Teradata legacy stages.

The Teradata legacy stages include:

- Teradata Enterprise stage — a parallel bulk data load and export solution using multiple FastLoad and FastExport sessions.
- Teradata Multiload stage — a bulk data load and export solution that uses the Teradata MultiLoad, TPump, and FastExport utilities.
- Teradata API stage — provides row-to-row read and write accesses to the Teradata database based on the SQL statements.
- Teradata Load stage — provides a bulk data load solution that uses the Teradata FastLoad utility.

The Teradata legacy stages are available on Information Server Version 7.5 and later.

This article does not cover the stored procedure (STP) stage and the Open Database Connectivity (ODBC) stage in detail. The IBM InfoSphere Information Server uses these stages to provide support for many database types:

- The STP stage provides the capability to call stored procedures in DB2, Oracle, Teradata, Sybase, and SQL server from DataStage jobs. It supports the stored procedures with input parameters, output parameters, or both. STP stage is the recommended solution to call the Teradata macros, stored procedures, scalar functions, and table functions.
- The ODBC stage provides the capability to use the ODBC driver to access various database systems, including Teradata.

Table 1 shows suggested options to select based on your use case. The concepts of the sparse and normal lookups are fully explained in the Look up Teradata data section.

Table 1. Teradata integration options

Use case	Suggested Solution	Legacy Options	Limitations
Low volume Teradata data read	Teradata Connector: immediate mode, SQL	Teradata API Stage	
Low volume data insert/update/ upsert/delete	Teradata Connector: Immediate mode, SQL Bulk mode, stream driver	Teradata API Stage Teradata Multiload Stage	
Realtime data Insert/update/ upsert/delete	Teradata Connector: Immediate mode, SQL Bulk mode, stream driver	Teradata API Stage Teradata Multiload stage	
Bulk data load to empty table	Teradata Connector: Bulk mode, load driver	Teradata Enterprise Stage Teradata Load Stage	Target table: No secondary indexes No referential integrity No triggers No multiset
Bulk load/update to existing Table Bulk data load to empty table with no unique secondary indexes	Teradata Connector Bulk mode, update driver	Teradata Multiload Stage	Target table: No Unique Secondary Indexes No referential integrity No triggers
Bulk data export	Teradata Connector Bulk mode, export driver	Teradata Enterprise Stage Teradata Multiload Stage	
Normal Lookup	Teradata Connector Bulk mode, export driver Immediate mode, SQL	Teradata Enterprise Stage Teradata Multiload Stage Teradata API Stage	
Sparse Lookup	Teradata Connector Immediate mode, SQL	Teradata API Stage	
Multiple input links and transaction support	Teradata Connector Immediate mode		
Calling Teradata stored procedures, macros, scalar functions, and table functions	Stored Procedure Stage (STP)		

Load data using the Teradata Connector

This section uses a sample ETL job to illustrate the steps to use the Teradata connector to load data into an empty Teradata table. Figure 2 shows the sample job. The job reads the orders from a flat file. It transforms and passes the source data to the Teradata connector named LoadDataUsingBulkLoad. The connector loads the data into an empty table named Orders using the Teradata parallel transporter load driver. Data records that violate the database constraints are rejected by the connector and forwarded to a flat file.

Figure 2. Load data into the Teradata Orders table

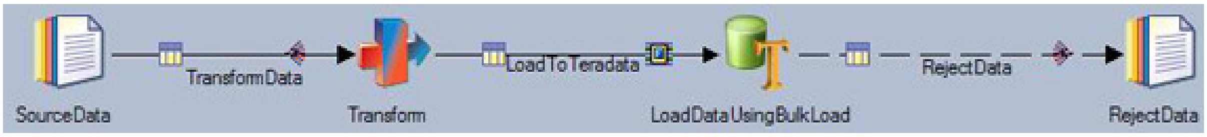
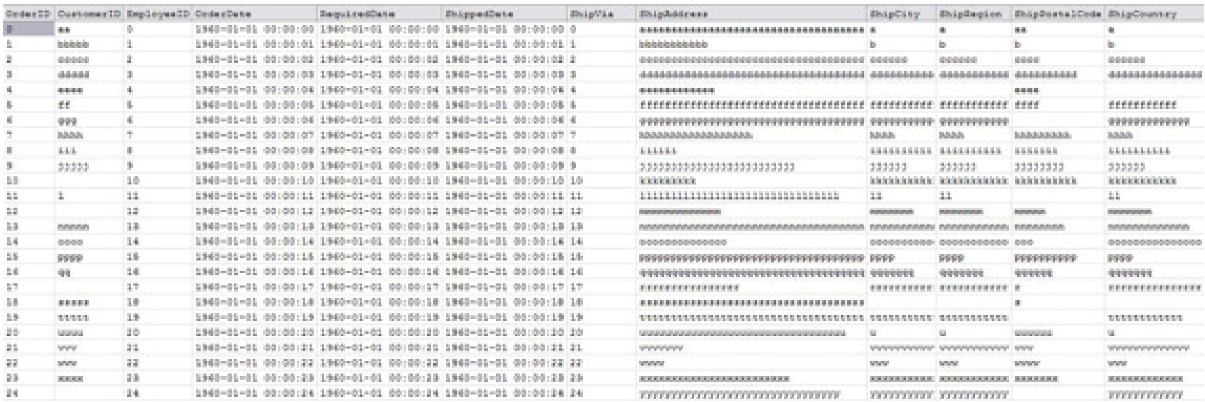


Figure 3 shows the sample source data.

Figure 3. Sample source data (5,000 rows)



OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	ShipAddress	ShipCity	ShipRegion	ShipPostalCode	ShipCountry
0	aa	0	1960-01-01 00:00:00	1960-01-01 00:00:00	1960-01-01 00:00:00	0	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	a	a	aa	a
1	bbbbb	1	1960-01-01 00:00:01	1960-01-01 00:00:01	1960-01-01 00:00:01	1	bbbbbbbbbbbb	b	b	b	b
2	ccccc	2	1960-01-01 00:00:02	1960-01-01 00:00:02	1960-01-01 00:00:02	2	cccccccccccccccccccccccccccccccccccccc	cccccc	cccccc	cccc	cccccc
3	ddddd	3	1960-01-01 00:00:03	1960-01-01 00:00:03	1960-01-01 00:00:03	3	dddddddddddddddddddddddddddddddddddddd	dddddddddd	ddddddddddd	dddddddddd	ddddddddddddddd
4	eeee	4	1960-01-01 00:00:04	1960-01-01 00:00:04	1960-01-01 00:00:04	4	eeeeeeeeeeeee			eeee	
5	ff	5	1960-01-01 00:00:05	1960-01-01 00:00:05	1960-01-01 00:00:05	5	ffffffffffffffffffffffffffffffffffffff	ffffffffff	fffffffffff	ffff	fffffffffff
6	ggg	6	1960-01-01 00:00:06	1960-01-01 00:00:06	1960-01-01 00:00:06	6	gggggggggggggggggggggggggggggggggggggg	ggggggggggg	ggggggggggg		ggggggggggggg
7	hhhh	7	1960-01-01 00:00:07	1960-01-01 00:00:07	1960-01-01 00:00:07	7	hhhhhhhhhhhhhhhhhhh	hhhh	hhhh	hhhhhhhhh	hhhh
8	iii	8	1960-01-01 00:00:08	1960-01-01 00:00:08	1960-01-01 00:00:08	8	iiiiii	iiiiiiiiii	iiiiiiiiii	iiiiiii	iiiiiiiiii
9	jjjjj	9	1960-01-01 00:00:09	1960-01-01 00:00:09	1960-01-01 00:00:09	9	jjjjjjjjjjjjjjjjjjjjjjjjjj	jjjjjj	jjjjjj	jjjjjjjj	jjjjjj
10		10	1960-01-01 00:00:10	1960-01-01 00:00:10	1960-01-01 00:00:10	10	kkkkkkkkk	kkkkkkkkkk	kkkkkkkkkkk	kkkkkkkkkk	kkkkkkkkkkk
11	l	11	1960-01-01 00:00:11	1960-01-01 00:00:11	1960-01-01 00:00:11	11	llllllllllllllllllllllllllllllllll	ll	ll		ll
12		12	1960-01-01 00:00:12	1960-01-01 00:00:12	1960-01-01 00:00:12	12	mmmmmmmmmmmmmm	mmmmmmm	mmmmmmm	mmmmm	mmmmmmm
13	nnnnn	13	1960-01-01 00:00:13	1960-01-01 00:00:13	1960-01-01 00:00:13	13	nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn	nnnnnnnnnnn	nnnnnnnnnnn	nnnnnnnn	nnnnnnnnnnnnn
14	oooo	14	1960-01-01 00:00:14	1960-01-01 00:00:14	1960-01-01 00:00:14	14	ooooooooooooooo	ooooooooooo	ooooooooooo	ooo	ooooooooooooooo
15	pppp	15	1960-01-01 00:00:15	1960-01-01 00:00:15	1960-01-01 00:00:15	15	pppppppppppppppppppppppppppppppppppppp	pppp	pppp	pppppppppp	pppp
16	qq	16	1960-01-01 00:00:16	1960-01-01 00:00:16	1960-01-01 00:00:16	16	qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq	qqqqqqq	qqqqqqq	qqqqqq	qqqqqqq
17		17	1960-01-01 00:00:17	1960-01-01 00:00:17	1960-01-01 00:00:17	17	rrrrrrrrrrrrrrrrr	rrrrrrrrrr	rrrrrrrrrrr	r	rrrrrrrrrrrrrrr
18	sssss	18	1960-01-01 00:00:18	1960-01-01 00:00:18	1960-01-01 00:00:18	18	ssssssssssssssssssssssssssssssssssssss			s	
19	ttttt	19	1960-01-01 00:00:19	1960-01-01 00:00:19	1960-01-01 00:00:19	19	tttttttttttttttttttttttttttttttttttttt	tttttttttt	ttttttttttt		tttttttttttt
20	uuuuu	20	1960-01-01 00:00:20	1960-01-01 00:00:20	1960-01-01 00:00:20	20	uuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuuu	u	u	uuuuuu	u
21	vvv	21	1960-01-01 00:00:21	1960-01-01 00:00:21	1960-01-01 00:00:21	21	vvvvvvv	vvvvvvvvvvv	vvvvvvvvvvvv	vvv	vvvvvvvvvvvvvv
22	www	22	1960-01-01 00:00:22	1960-01-01 00:00:22	1960-01-01 00:00:22	22	wwww	www	www	wwww	www
23	xxxx	23	1960-01-01 00:00:23	1960-01-01 00:00:23	1960-01-01 00:00:23	23	xxxxxxxxxxxxxxxxxxxxxxxxxx	xxxxxxxxxxx	xxxxxxxxxxx	xxxxxxx	xxxxxxxxxxxx
24		24	1960-01-01 00:00:24	1960-01-01 00:00:24	1960-01-01 00:00:24	24	yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy	yyyyyyyyyy	yyyyyyyyyyy		yyyyyyyyyyyy

(See a larger version of Figure 3.)

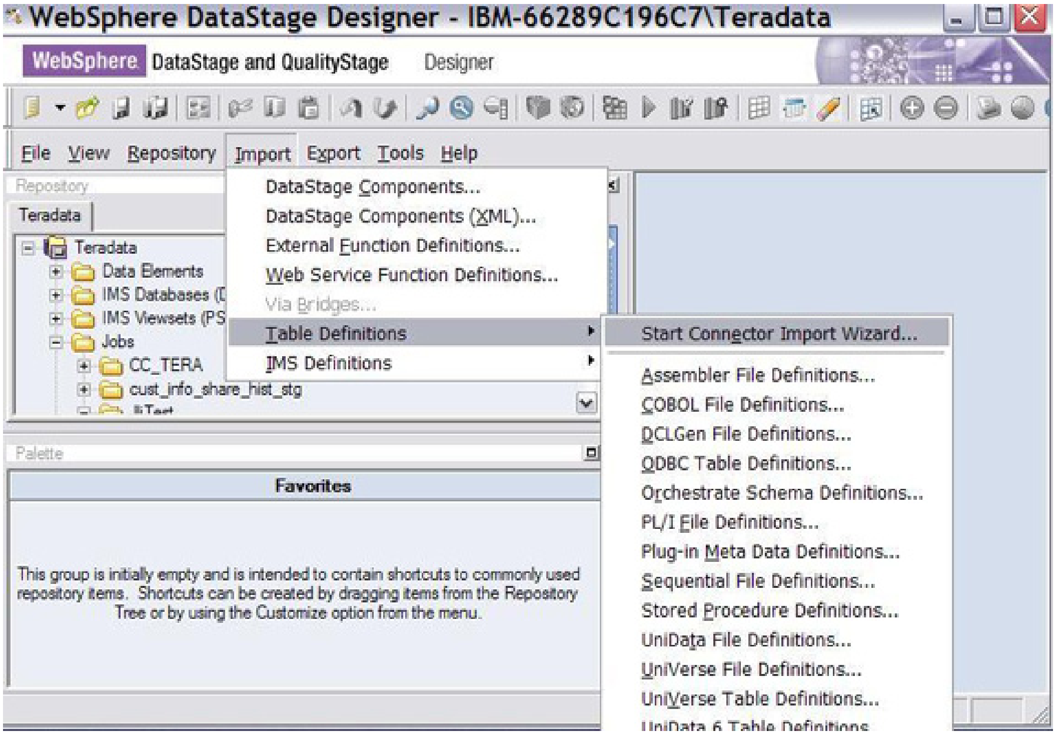
Listing 1 shows the SQL for creating the Teradata database Orders table.

Listing 1. SQL to create the Orders table

```
CREATE SET TABLE Orders ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL,
     CHECKSUM = DEFAULT
     (
      OrderID INTEGER NOT NULL,
      CustomerID VARCHAR(5) CHARACTER SET LATIN CASESPECIFIC,
      EmployeeID INTEGER,
      OrderDate TIMESTAMP(0),
      RequiredDate TIMESTAMP(0),
      ShippedDate TIMESTAMP(0),
      ShipVia INTEGER,
      ShipAddress VARCHAR(60) CHARACTER SET LATIN CASESPECIFIC,
      ShipCity VARCHAR(15) CHARACTER SET LATIN CASESPECIFIC,
      ShipRegion VARCHAR(15) CHARACTER SET LATIN CASESPECIFIC,
      ShipPostalCode VARCHAR(10) CHARACTER SET LATIN CASESPECIFIC,
      ShipCountry VARCHAR(15) CHARACTER SET LATIN CASESPECIFIC)
UNIQUE PRIMARY INDEX ( OrderID );
```

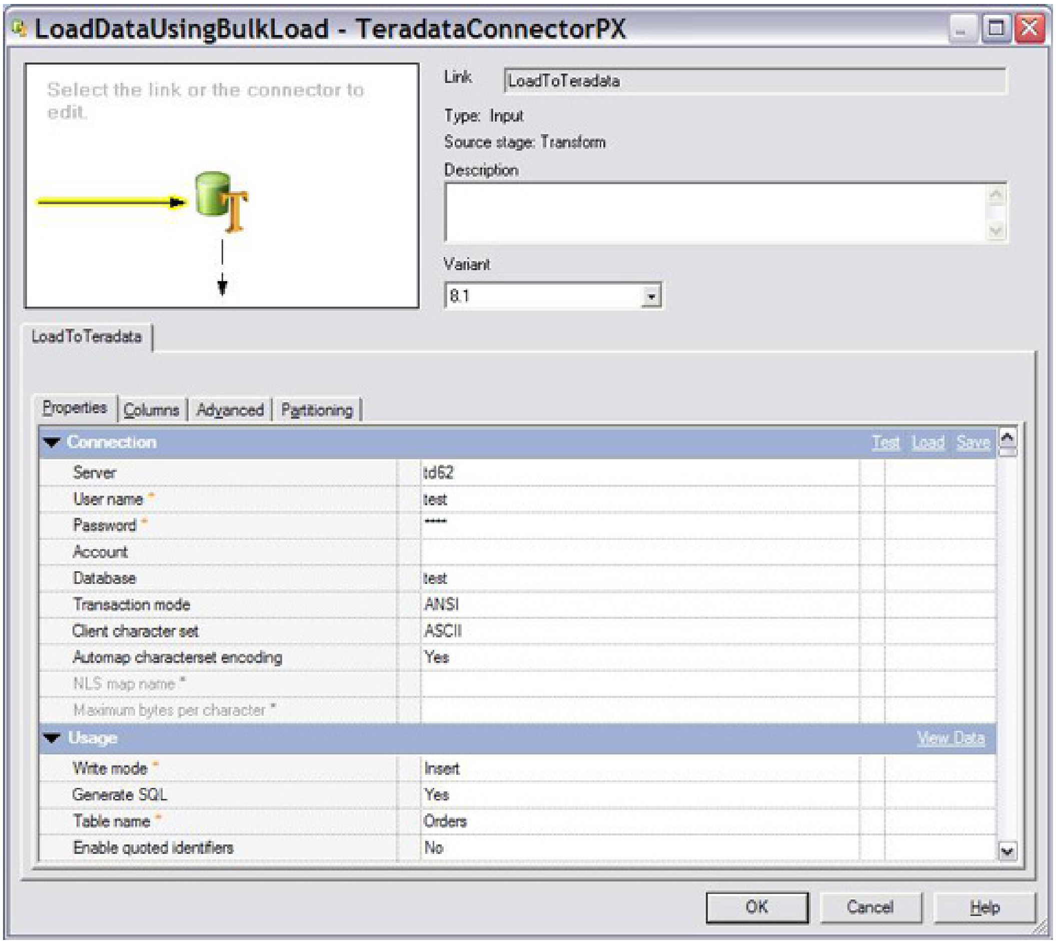
Two main steps are required to set up the LoadDataUsingBulkLoad Teradata connector for the bulk data load operation:

1. Launch the connector importer wizard, as shown in Figure 4, and import the definition for the Orders table.
Figure 4. Launch the Connector Import Wizard


WebSphere DataStage Designer - IBM-66289C196C7\Teradata
WebSphere DataStage and QualityStage
Designer
File
View
Repository
Import
Export
Tools
Help
Repository
Teradata
Teradata
Data Elements
IMS Databases (D
IMS Viewsets (PS
Jobs
CC_TERA
cust_info_share_hist_stg
DataStage Components...
DataStage Components (XML)...
External Function Definitions...
Web Service Function Definitions...
Via Bridges...
Table Definitions
IMS Definitions
Start Connector Import Wizard...
Assembler File Definitions...
COBOL File Definitions...
DCLGen File Definitions...
ODBC Table Definitions...
Orchestrate Schema Definitions...
PL/I File Definitions...
Plug-in Meta Data Definitions...
Sequential File Definitions...
Stored Procedure Definitions...
UniData File Definitions...
UniVerse File Definitions...
UniVerse Table Definitions...
UniData 6 Table Definitions
Palette
Favorites

This group is initially empty and is intended to contain shortcuts to commonly used repository items. Shortcuts can be created by dragging items from the Repository Tree or by using the Customize option from the menu.

2. Open the **Properties** tab for the connector as shown in Figure 5, and set up the Teradata data operation.
Figure 5. Teradata Connector property editor



Select the link or the connector to edit.

Link: LoadToTeradata

Type: Input

Source stage: Transform

Description:

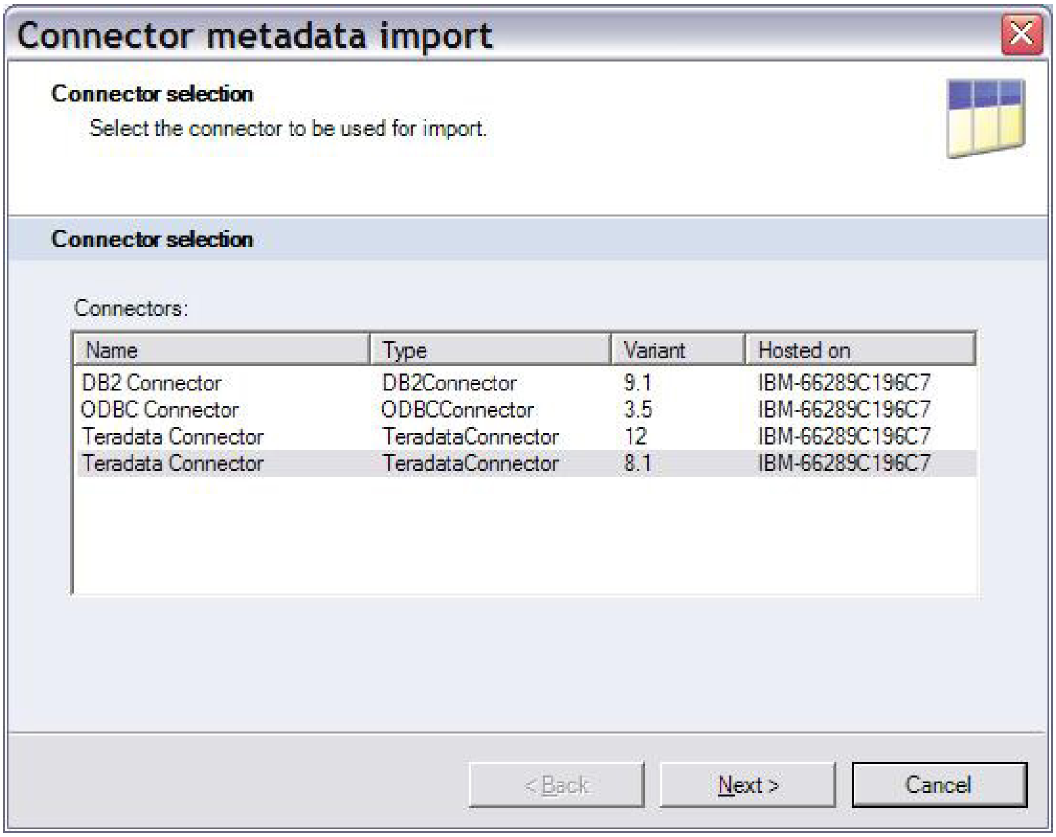
Variant: 8.1

Connection	
Server	td62
User name *	test
Password *	*****
Account	
Database	test
Transaction mode	ANSI
Client character set	ASCII
Automap characterset encoding	Yes
NLS map name *	
Maximum bytes per character *	
Usage	
Write mode *	Insert
Generate SQL	Yes
Table name *	Orders
Enable quoted identifiers	No

Import Teradata table definition

This section contains screenshots that illustrate the steps to import the definition of the Orders database table.

1. As shown in Figure 6, select the Teradata Connector to import Teradata table schemas. The Teradata connector variant 8.1 supports the Teradata TTU 8.1 and 8.2. The variant 12 supports the Teradata TTU 12 and 13. This sample uses TTU 8.2. As shown in Figures 5 and 6, these example jobs select the Teradata connector variant 8.1.
Figure 6. Select the Teradata connector for Metadata import



Connector selection

Select the connector to be used for import.

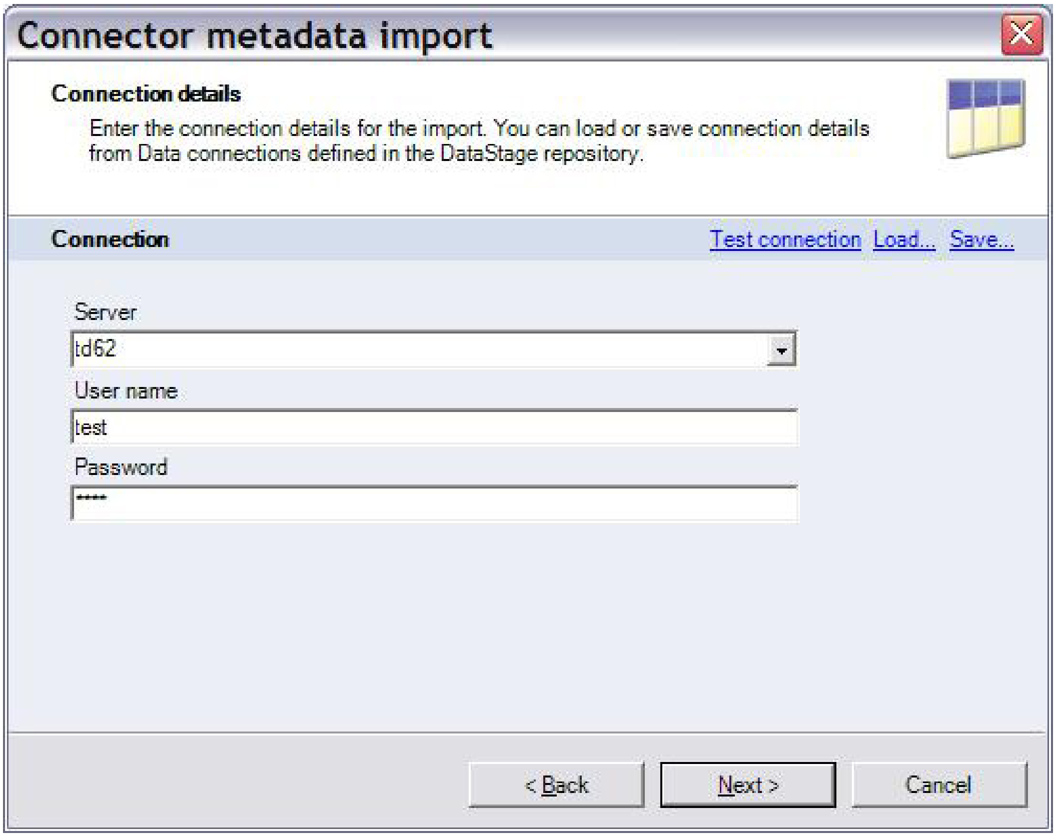
Connector selection

Connectors:

Name	Type	Variant	Hosted on
DB2 Connector	DB2Connector	9.1	IBM-66289C196C7
ODBC Connector	ODBCConnector	3.5	IBM-66289C196C7
Teradata Connector	TeradataConnector	12	IBM-66289C196C7
Teradata Connector	TeradataConnector	8.1	IBM-66289C196C7

2. As shown in Figure 7, specify the Teradata Director Program ID (TDPID), user name, and password so that the import wizard can connect to the Teradata database. The TDPID is an alias that the Teradata client applications use to connect to the Teradata database. The TDPID parameters are defined in the machine hosts file (/etc/hosts for UNIX®, or \windows\system32\drivers\etc\hosts for Windows®).

Figure 7. Enter the connection details



Connector metadata import

Connection details

Enter the connection details for the import. You can load or save connection details from Data connections defined in the DataStage repository.

Connection

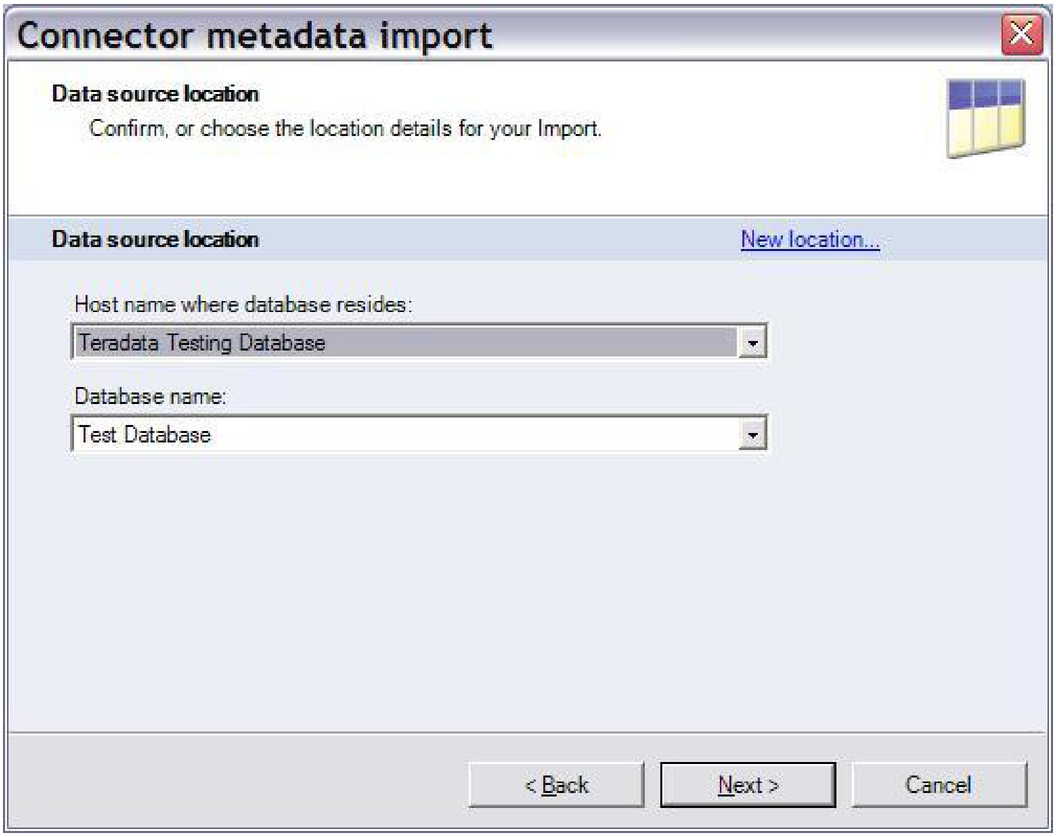
Server: td62

User name: test

Password: ****

3. As shown in Figure 8, specify where the table is imported from. This information is mainly used by the DataStage to track the original source for the table definition.

Figure 8. Enter the location details



Connector metadata import

Data source location

Confirm, or choose the location details for your Import.

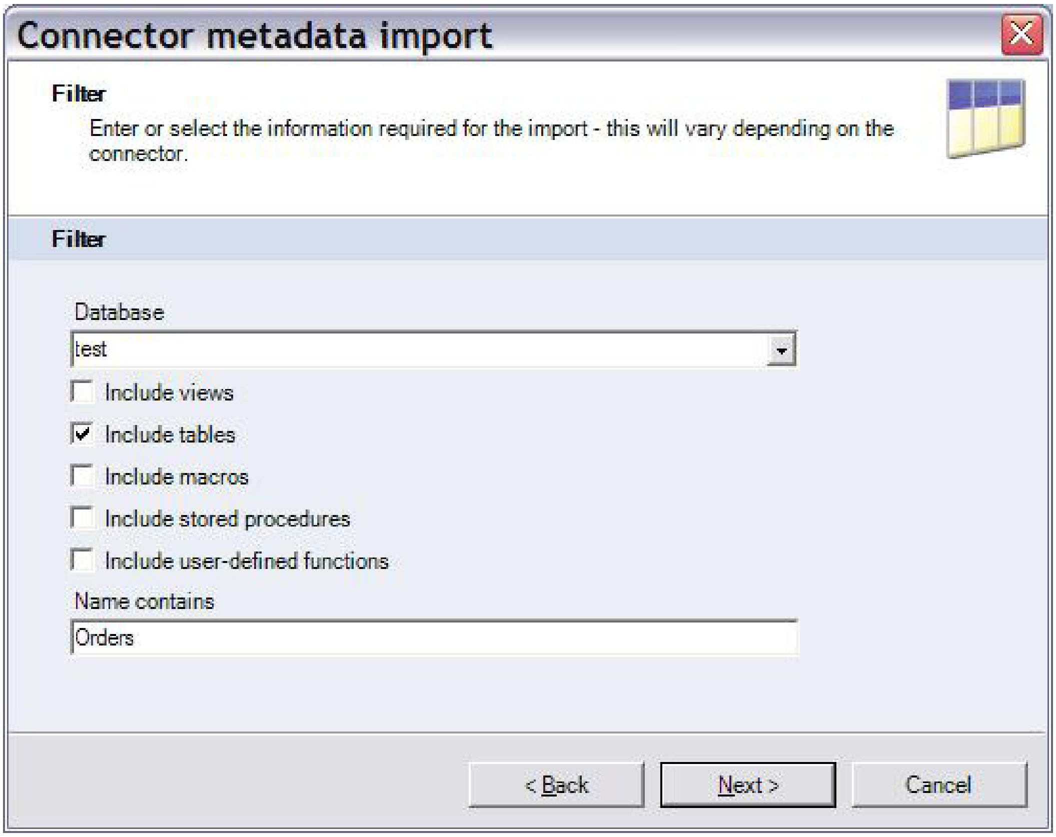
Data source location

Host name where database resides: Teradata Testing Database

Database name: Test Database

4. As shown in Figure 9, provide filters to narrow the table search result.

Figure 9. Filter a list of tables by database and table name



Connector metadata import

Filter

Enter or select the information required for the import - this will vary depending on the connector.

Filter

Database

test

□ Include views

☑ Include tables

□ Include macros

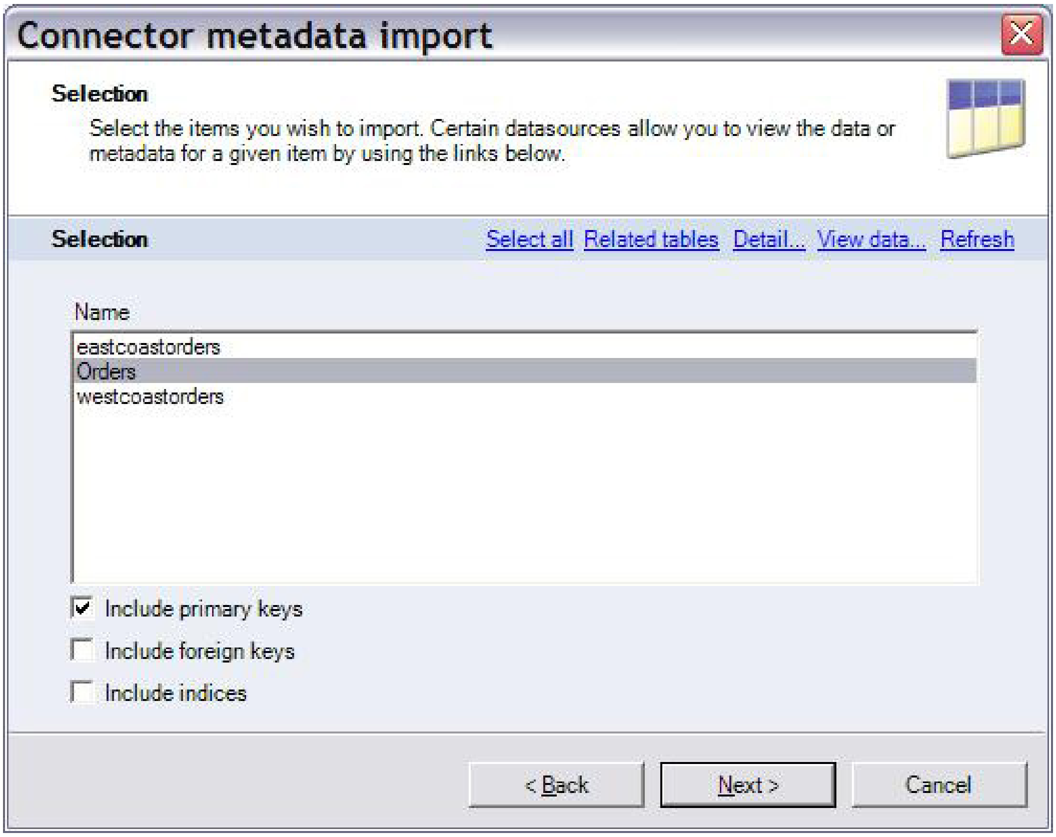
□ Include stored procedures

□ Include user-defined functions

Name contains

Orders

5. As shown in Figure 10, all the tables matching the filter conditions set in Figure 9 are returned. Select the Orders table for the import operation.
   Figure 10. Select a Teradata table



Connector metadata import

Selection

Select the items you wish to import. Certain datasources allow you to view the data or metadata for a given item by using the links below.

Selection

Name

- eastcoastorders
- Orders
- westcoastorders

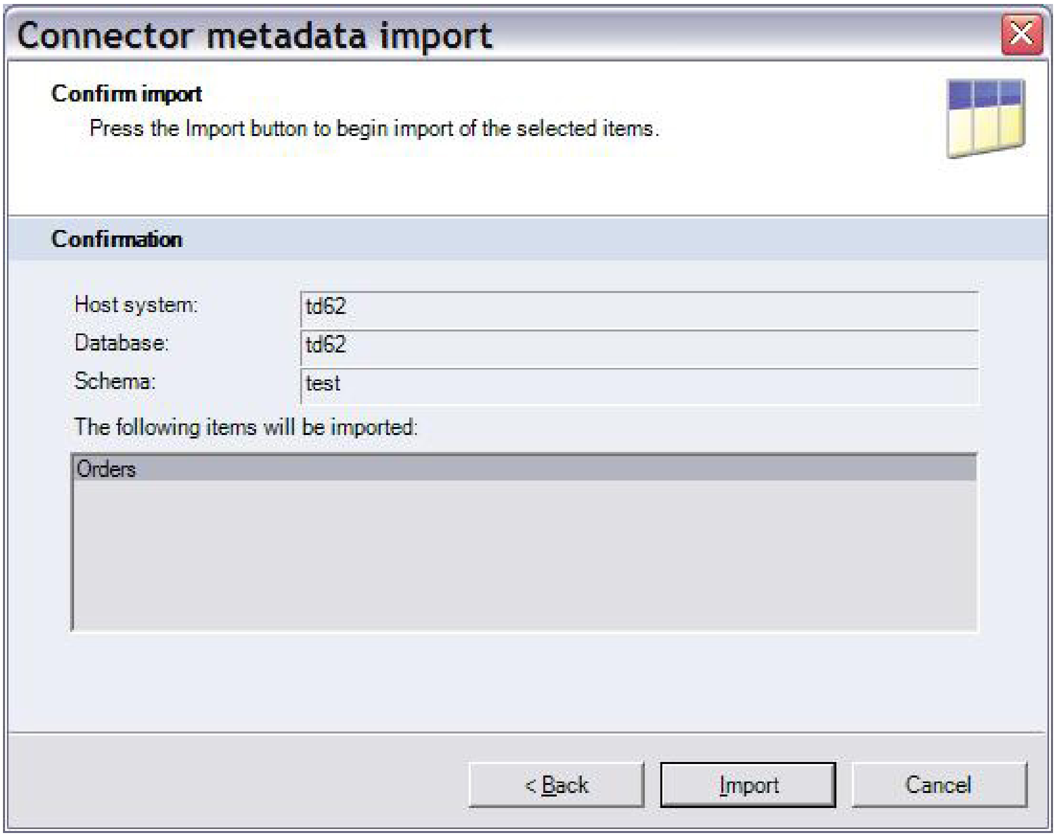
☑ Include primary keys

□ Include foreign keys

□ Include indices

6. Figure 11 shows the confirmation screen for the import operation. Click the **Import** button to start the import operation. In this example, a DataStage table definition named test.Orders is created in the DataStage repository based on the database table schema for the Orders table.

Figure 11. Confirm the import operation



Connector metadata import

Confirm import

Press the Import button to begin import of the selected items.

Confirmation

Host system: td62

Database: td62

Schema: test

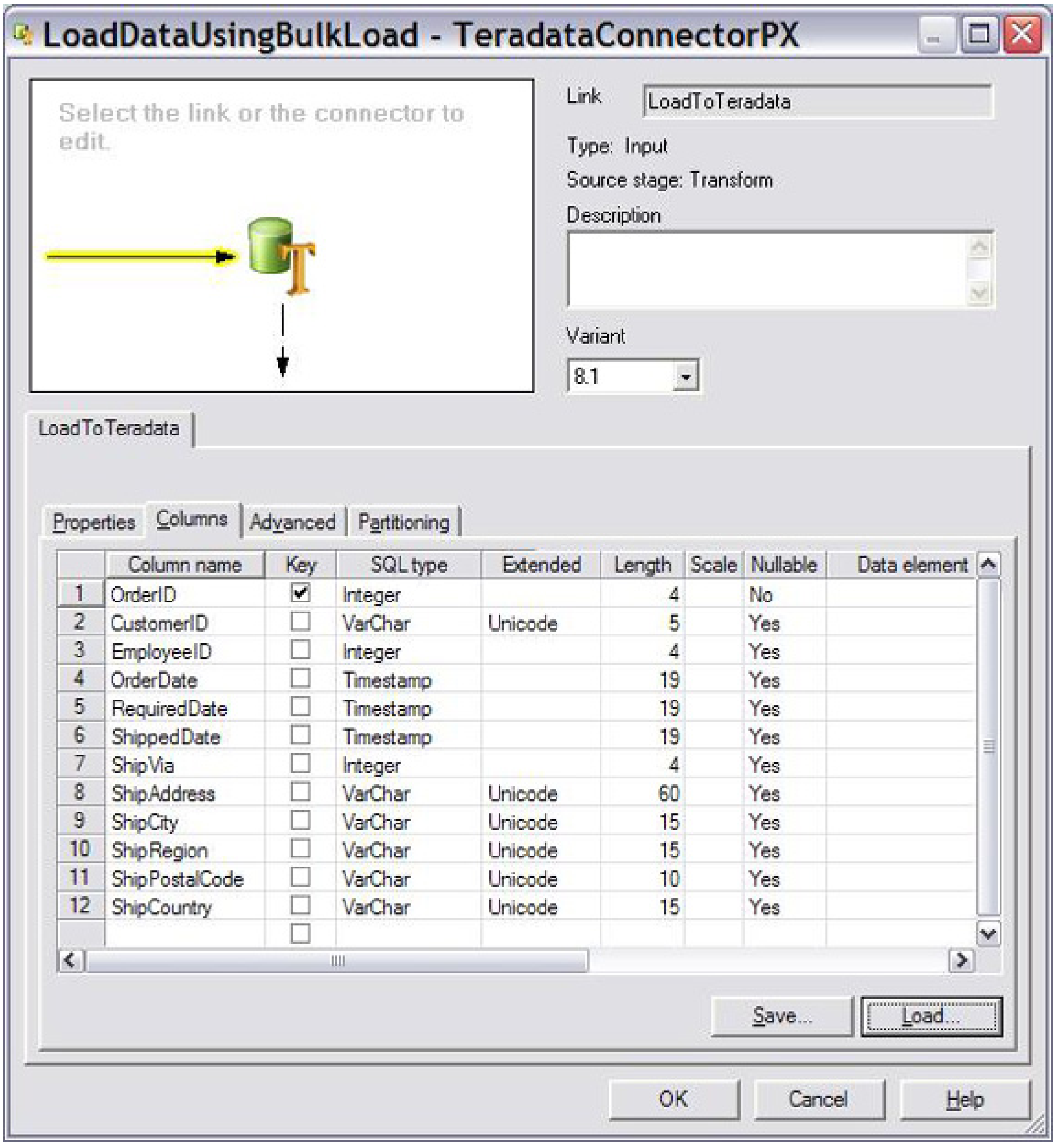
The following items will be imported:

Orders

Set up the bulk load operation on the Property Editor

This section contains screenshots that illustrate the steps to define the bulk load operation using the Teradata Parallel Transport load driver.

1. As shown in Figure 12, open the **Columns** tab for the connector, select the **OrderID** column as the key, and click the **Load** button to load the test.Orders table definition from the DataStage repository to the LoadDataUsingBulkLoad connector.
   Figure 12. Load column definition



LoadDataUsingBulkLoad - TeradataConnectorPX
Select the link or the connector to edit.
Properties
Columns
Advanced
Partitioning
Save...
Load...
OK
Cancel
Help

Link: LoadToTeradata

Type: Input

Source stage: Transform

Description

Variant: 8.1

LoadToTeradata

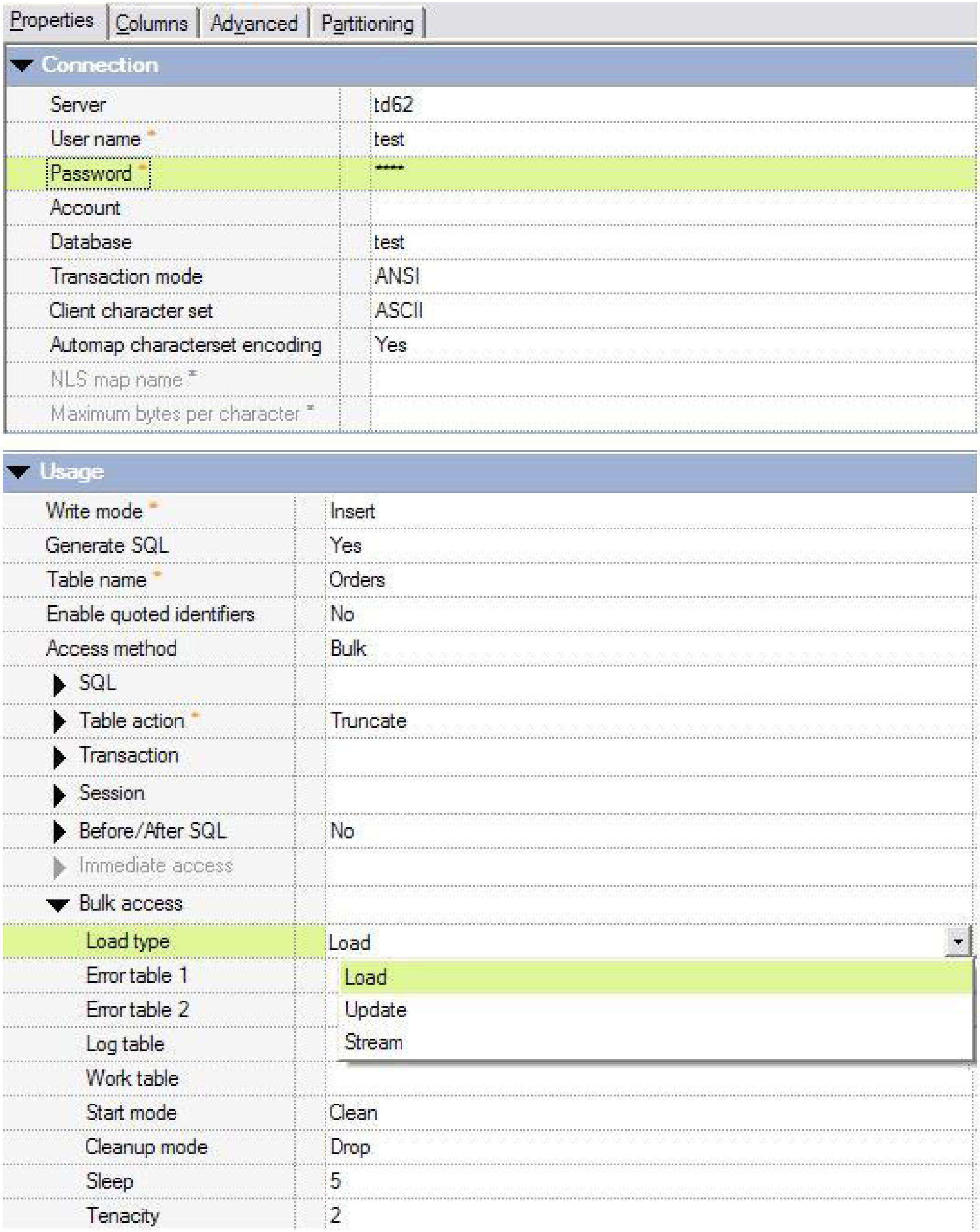
	Column name	Key	SQL type	Extended	Length	Scale	Nullable	Data element
1	OrderID	☑	Integer		4		No	
2	CustomerID	□	VarChar	Unicode	5		Yes	
3	EmployeeID	□	Integer		4		Yes	
4	OrderDate	□	Timestamp		19		Yes	
5	RequiredDate	□	Timestamp		19		Yes	
6	ShippedDate	□	Timestamp		19		Yes	
7	ShipVia	□	Integer		4		Yes	
8	ShipAddress	□	VarChar	Unicode	60		Yes	
9	ShipCity	□	VarChar	Unicode	15		Yes	
10	ShipRegion	□	VarChar	Unicode	15		Yes	
11	ShipPostalCode	□	VarChar	Unicode	10		Yes	
12	ShipCountry	□	VarChar	Unicode	15		Yes	
		□						

2. As shown in Figure 13, return to the **Properties** tab of the connector and specify the following parameters for the Teradata load operation:

   - The connection details: Teradata Program Director ID (TPDID), user name, password, and database.
   - The Teradata client character set that is used to communicate with the Teradata server.
   - The target database table name, auto-generate SQL, and table action. The connector will create the insert SQL statement based on the target table name of Orders and the column definition selected in Figure 12. Select the **Truncate** table action to delete all the existing rows in the Orders table before the data load operation starts.

- The bulk access method and driver. The load driver is selected to load data into the empty Orders table.
- The sleep and tenacity settings. Teradata database limits the combined number of FastLoad, MultiLoad, and FastExport tasks that are allowed to run concurrently. This is controlled by the MaxLoadTasks and MaxLoadAWT DBS control fields. Normally the limit is from 5 to 15. The sleep and tenacity settings affect how the connector retries to connect to the database when the limit is exceeded. The sleep value specifies the interval between retries in minutes. The tenacity value specifies the timeout for the login retry in hours.

Figure 13. Enter connection details and set up the bulk load operation



Connection

Server	td62
User name *	test
Password *	****
Account	
Database	test
Transaction mode	ANSI
Client character set	ASCII
Automap characterset encoding	Yes
NLS map name *	
Maximum bytes per character *	

Usage

Write mode *	Insert
Generate SQL	Yes
Table name *	Orders
Enable quoted identifiers	No
Access method	Bulk
SQL	
Table action *	Truncate
Transaction	
Session	
Before/After SQL	No
Immediate access	
Bulk access	
Load type	Load
Error table 1	
Error table 2	
Log table	
Work table	
Start mode	Clean
Cleanup mode	Drop
Sleep	5
Tenacity	2

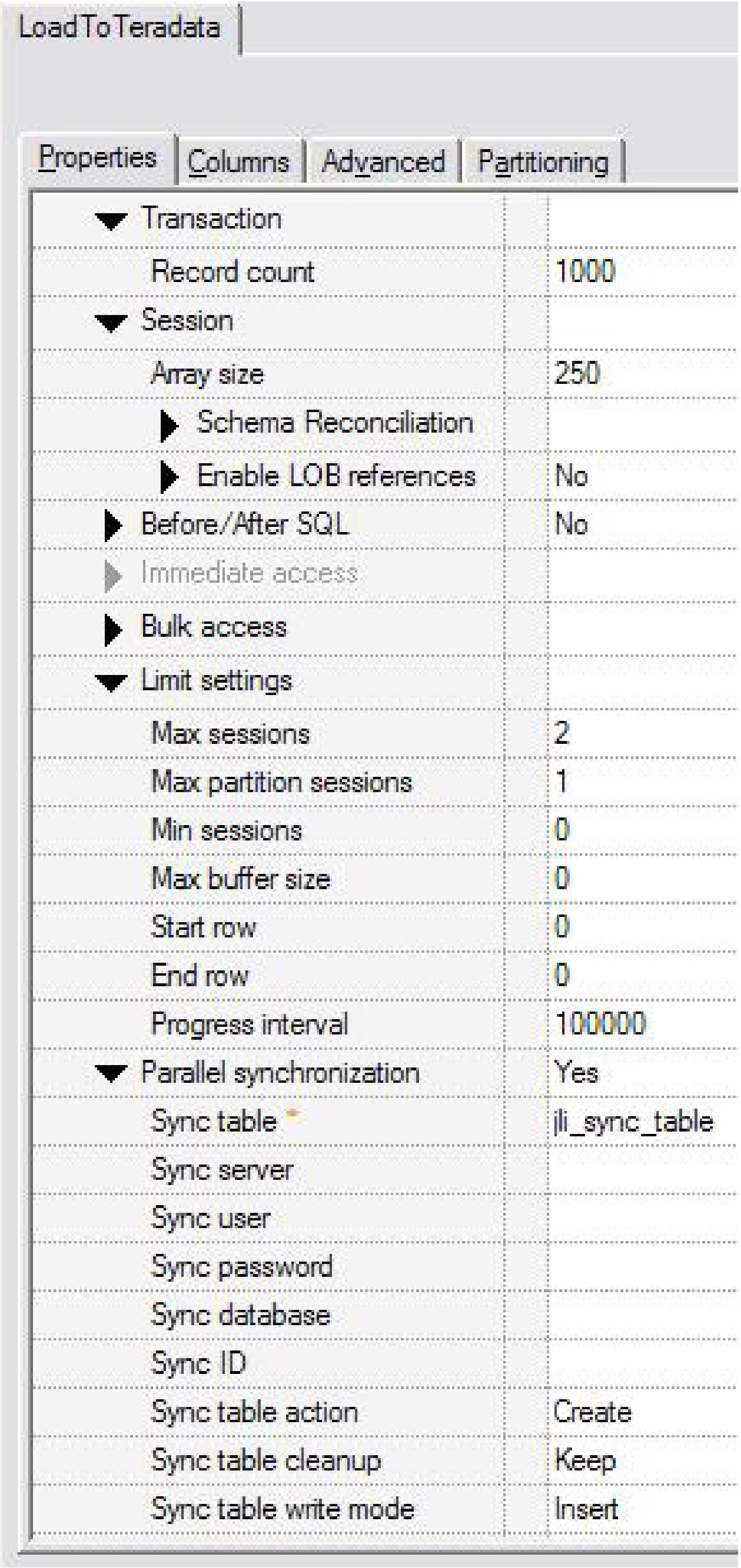
3. As shown in Figure 14, continue defining parameter values in the **Properties** tab of the connector for the Teradata load operation:

   - The Record Count for setting the checkpoint. The connector supports the checkpoint and restart feature in the Teradata parallel Transport. A checkpoint defines a point during the loading process at which the database has successfully processed a specified number of records.

If an error occurs after a checkpoint, the loading process can be restarted from the rows following the checkpoint. This example sets the checkpoint for every 1,000 rows.

- The Sync table and action. The connector supports performing the data load operation using multiple processes running on multiple computing nodes. The connector is dependent on the Teradata parallel transport interface that requires multiple processes to be synchronized at various points during the load process. This example specifies that the connector creates and uses the jli_sync_table table for process synchronization.
- The maximum sessions and maximum partition sessions. These two parameters specify the maximum number of sessions used for the data load operation and the maximum number of session used by each loading process. This example runs on two computing nodes and the targeted Teradata database has two AMPs. The values shown in Figure 14 specify that two processes use two sessions to load data into the database table.
- The array size. This parameter specifies how many rows a connector should cache before the data is sent to the load driver. The load driver will use the data from the connector and build 64k buffer to be sent to the Teradata database.

Figure 14. Restart, parallel load and synchronization, and others



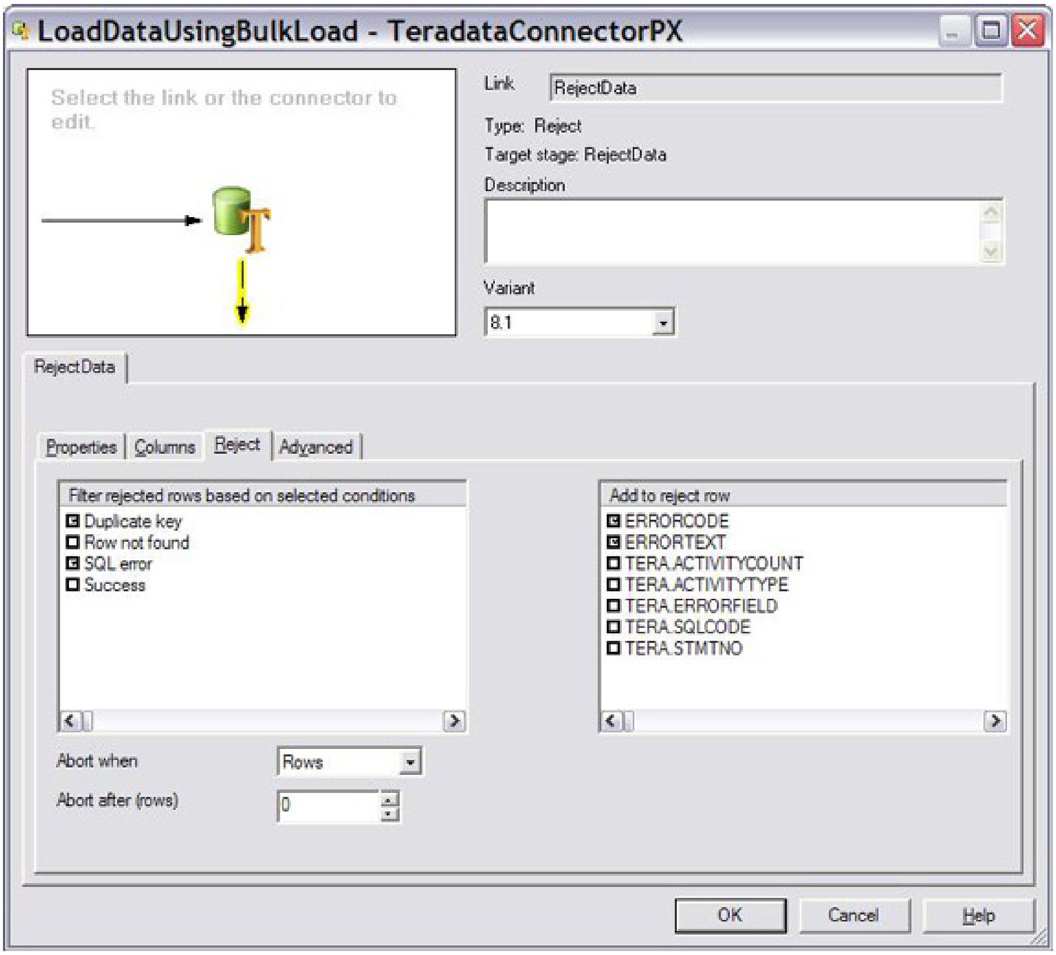
▼ Transaction	
Record count	1000
▼ Session	
Array size	250
▶ Schema Reconciliation	
▶ Enable LOB references	No
▶ Before/After SQL	No
▶ Immediate access	
▶ Bulk access	
▼ Limit settings	
Max sessions	2
Max partition sessions	1
Min sessions	0
Max buffer size	0
Start row	0
End row	0
Progress interval	100000
▼ Parallel synchronization	Yes
Sync table *	jli_sync_table
Sync server	
Sync user	
Sync password	
Sync database	
Sync ID	
Sync table action	Create
Sync table cleanup	Keep
Sync table write mode	Insert

4. Open the **Rejects** tab to specify how to deal with the error conditions. As shown in Figure 15, specify the following:

   - Select **Duplicate key** and **SQL errors** so that records that cause these errors are sent to the reject link.
   - Select **ERRORCODE** and **ERRORTEXT** to add these columns to each rejected data record to indicate why a record is rejected. The other selections do not apply to the load driver.

   One special filter condition that is not used in this example is the Success condition. The Success filter is designed to forward successfully processed records to the next stage for further processing.

   Figure 15. Set up the reject link



LoadDataUsingBulkLoad - TeradataConnectorPX

Link: RejectData

Type: Reject

Target stage: RejectData

Variant: 8.1

Filter rejected rows based on selected conditions

☑ Duplicate key
☐ Row not found
☑ SQL error
☐ Success

Add to reject row

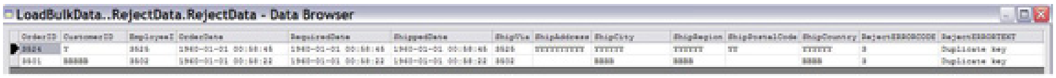
☑ ERRORCODE
☑ ERRORTEXT
☐ TERA.ACTIVITYCOUNT
☐ TERA.ACTIVITYTYPE
☐ TERA.ERRORFIELD
☐ TERA.SQLCODE
☐ TERA.STMTNO

Abort when: Rows

Abort after (rows): 0

5. As shown in Figure 16, the bulk load operation results in two of the sample records being sent to the reject link because they violate the unique primary key constraint.

   Figure 16. Rejected records



LoadBulkData..RejectData.RejectData - Data Browser

OrderID	CustomerID	EmployeeI	OrderDate	RequiredDate	ShippedDate	ShipVia	ShipAddress	ShipCity	ShipRegion	ShipPostalCode	ShipCountry	RejectERRORCODE	RejectERRORTEXT
3524	T	3525	1960-01-01 00:58:45	1960-01-01 00:58:45	1960-01-01 00:58:45	3525	TTTTTTTTTT	TTTTTT	TTTTTT	TT	TTTTTT	3	Duplicate key
3601	88888	3602	1960-01-01 00:58:22	1960-01-01 00:58:22	1960-01-01 00:58:22	3602		8888	8888		8888	3	Duplicate key

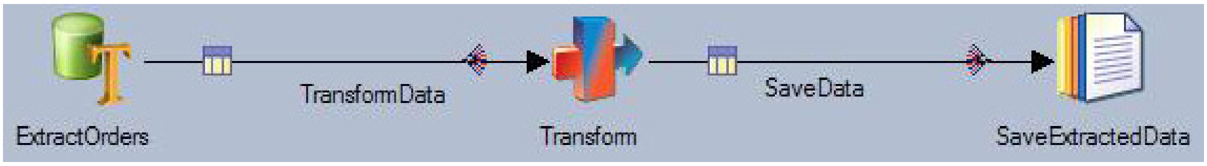
(See a larger version of Figure 16.)

Extract data using the Teradata Connector

This section uses a sample ETL job to illustrate the steps to extract data from the Teradata table named Orders. Figure 17 shows the sample job, which uses the immediate access mode. The job uses the Teradata connector named ExtractOrders to read the orders from the Orders database. The job transforms and passes the extracted data to the sequential file stage named SaveExtractedData.

The ExtractOrders connector uses the same table definition shown in Figure 12 and the same connection details shown in Figure 13.

Figure 17. Extract data from the Teradata Orders table

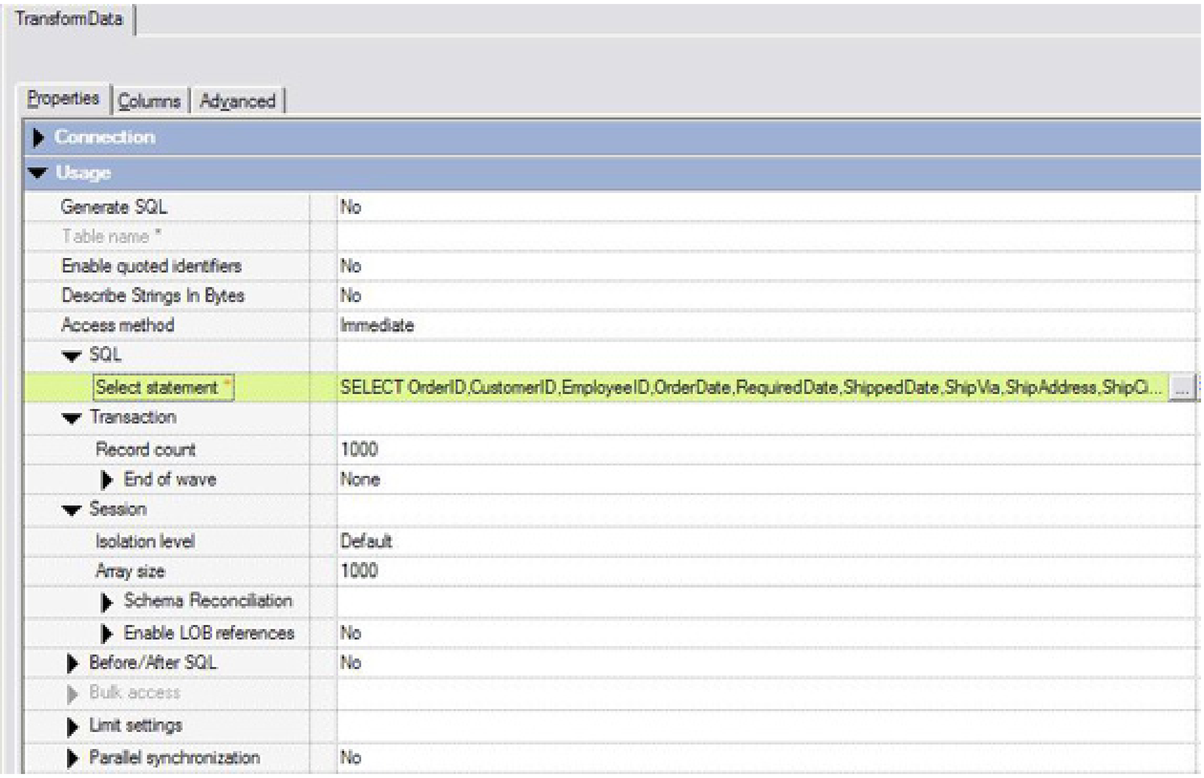


As shown in Figure 18, specify the following parameters for the data extraction operation:

- The immediate access method to run the SQL through the Teradata DBC/SQL partition. This job can run only in sequential mode on the DataStage conductor node and it is suited for small data extraction. To support the parallel extraction of large amount of data, the connector needs to be configured to use the bulk access method and the Teradata Parallel Transporter export driver.
- The select statement. The connector can generate the SQL using the table name and column definitions as shown in Figure 13. In this example, the SQL statement is manually entered.
- The record count. The record count is often used in combination with the End of Wave feature. You can use the End of Wave feature to divide the input/output records into a number of small transactions, or units of work. This example does no use the End of Wave feature, and the record count does not affect the data extraction operation.
- The array size. The array size is mainly designed for the connector to cache the input records for immediate and bulk load operations. It has no effect on this data extraction operation. The connector sets the maximum parcel size for the communication between the Teradata database and

connector to 64k or 1MB if the Teradata database server supports the 1MB parcel with the four-byte alternative parcel header (APH).

Figure 18. Set up the data extraction operation



Generate SQL	No
Table name *	
Enable quoted identifiers	No
Describe Strings In Bytes	No
Access method	Immediate
SQL	
Select statement *	SELECT OrderID,CustomerID,EmployeeID,OrderDate,RequiredDate,ShippedDate,ShipVia,ShipAddress,ShipCi...
Transaction	
Record count	1000
End of wave	None
Session	
Isolation level	Default
Array size	1000
Schema Reconciliation	
Enable LOB references	No
Before/After SQL	No
Bulk access	
Limit settings	
Parallel synchronization	No

Look up Teradata data

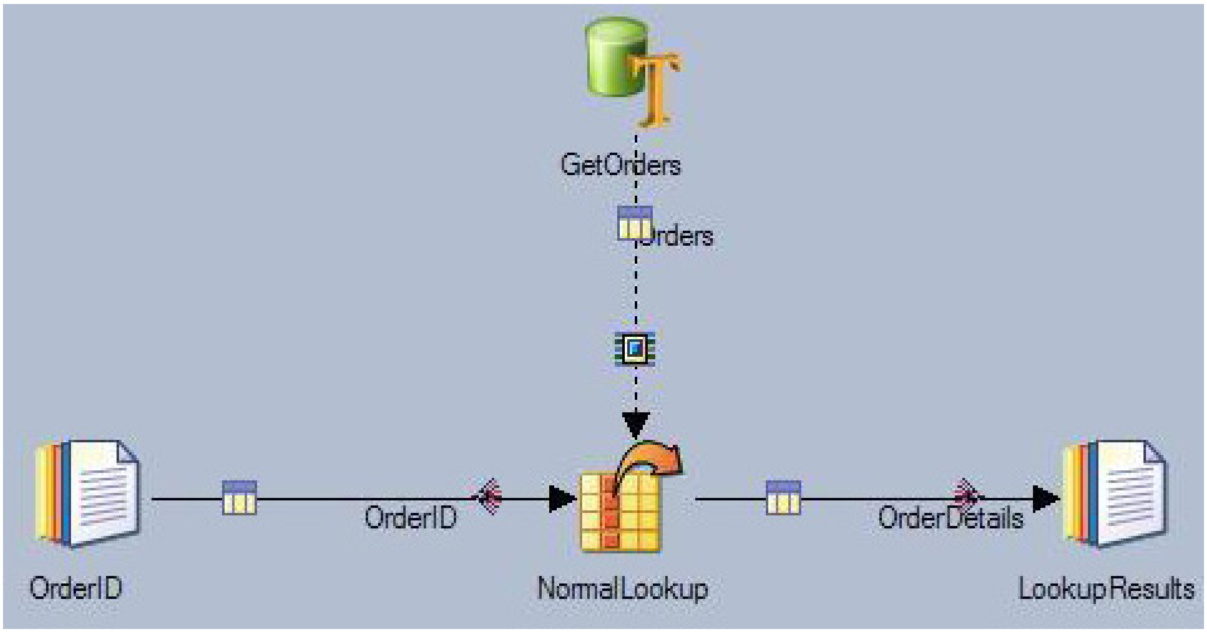
This section uses two ETL jobs to illustrate the steps to look up the Teradata data based on the input records. The examples query the order details based on the input order IDs. The following sections explain each of the two kinds of lookups that the DataStage supports: normal and sparse.

Normal lookup

For normal lookup, all the referenced data is retrieved once from the target database and cached in memory or on disk. For each input record, the cached-referenced data is cross-checked to find the result.

Figure 19 shows the lookup stage and the Teradata connector of a sample job to perform a normal lookup. The Teradata connector performs a full table query on the Orders table and sends the query result to the lookup stage named NormalLookup. The lookup stage caches the query result and performs the lookup operations on the cached order details based on the order IDs from the OrderID input link. The results are sent to the output link named OrderDetails. This job requires one full table database query.

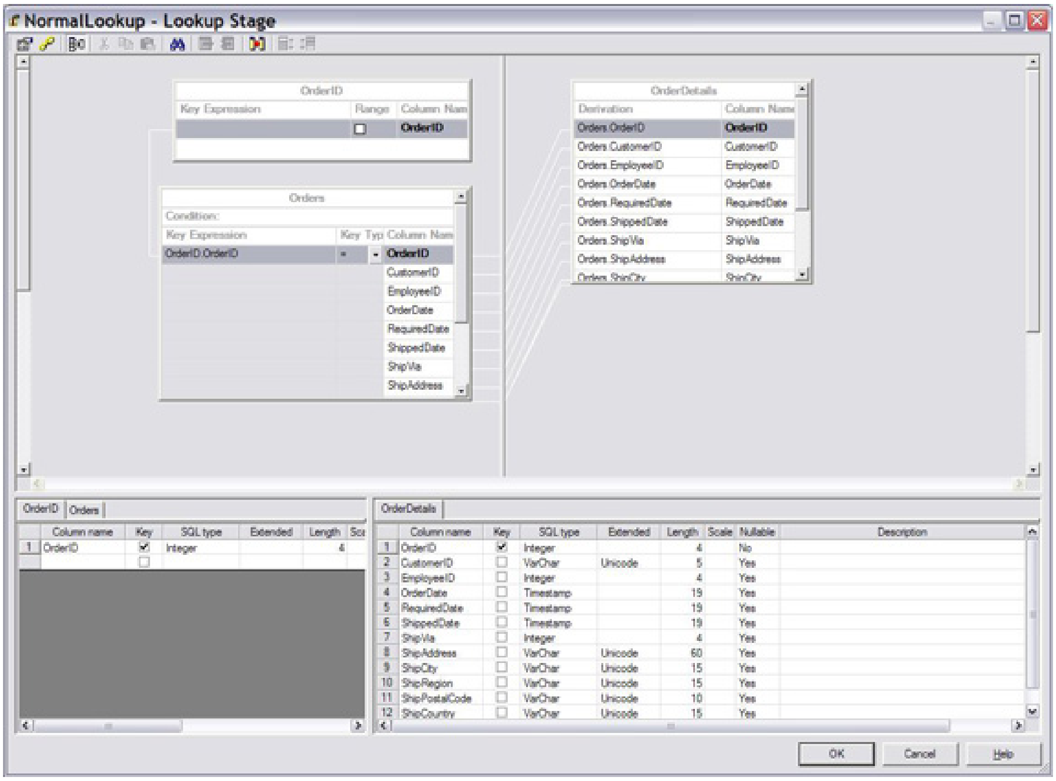
Figure 19. DataStage job to perform normal lookup



Two main steps are required to perform a normal lookup:

1. As shown in Figure 20, set up the lookup stage to perform the normal lookup.

Figure 20. Set up the lookup stage for normal lookup



OrderID

Key Expression	Range	Column Nam
	□	OrderID

Orders

Condition:

Key Expression	Key Typ	Column Nam
OrderID.OrderID	=	OrderID
		CustomerID
		EmployeeID
		OrderDate
		RequiredDate
		ShippedDate
		ShipVia
		ShipAddress

OrderDetails

Derivation	Column Name
Orders.OrderID	OrderID
Orders.CustomerID	CustomerID
Orders.EmployeeID	EmployeeID
Orders.OrderDate	OrderDate
Orders.RequiredDate	RequiredDate
Orders.ShippedDate	ShippedDate
Orders.ShipVia	ShipVia
Orders.ShipAddress	ShipAddress
Orders.ShipCity	ShipCity

OrderID | Orders

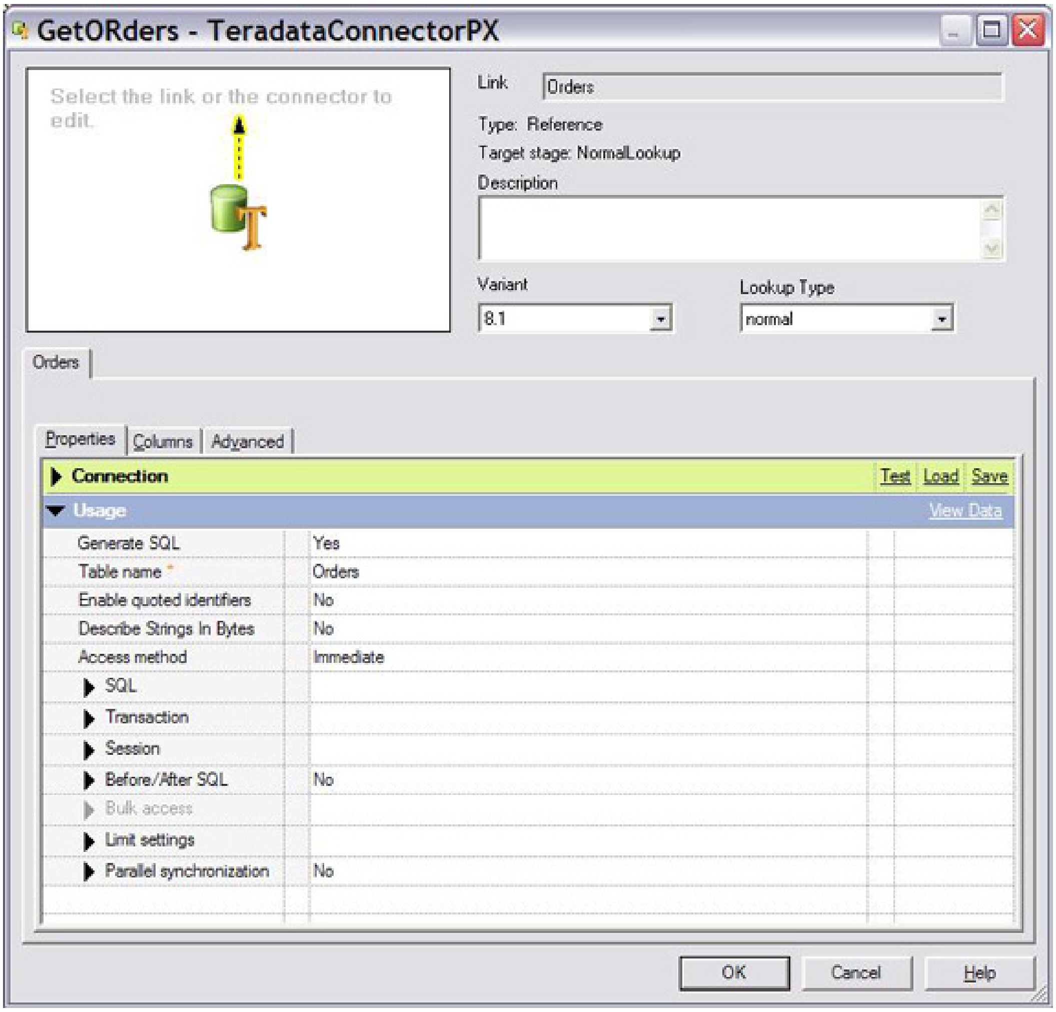
	Column name	Key	SQL type	Extended	Length	Sca
1	OrderID	☑	Integer		4	
		□				

OrderDetails

	Column name	Key	SQL type	Extended	Length	Scale	Nullable	Description
1	OrderID	☑	Integer		4		No	
2	CustomerID	□	VarChar	Unicode	5		Yes	
3	EmployeeID	□	Integer		4		Yes	
4	OrderDate	□	Timestamp		19		Yes	
5	RequiredDate	□	Timestamp		19		Yes	
6	ShippedDate	□	Timestamp		19		Yes	
7	ShipVia	□	Integer		4		Yes	
8	ShipAddress	□	VarChar	Unicode	60		Yes	
9	ShipCity	□	VarChar	Unicode	15		Yes	
10	ShipRegion	□	VarChar	Unicode	15		Yes	
11	ShipPostalCode	□	VarChar	Unicode	10		Yes	
12	ShipCountry	□	VarChar	Unicode	15		Yes	

2. As shown in Figure 21, specify the following parameters to set up the Teradata connector for normal lookup:
   - The normal lookup type.
   - The immediate access method. The bulk access method can also be used in the normal lookup.
   - The Orders target table and auto-generate SQL. The connector generates the query SQL at runtime based on the target table and the column definitions.

Figure 21. Set up the Teradata connector for normal lookup



Link: Orders
Type: Reference
Target stage: NormalLookup
Description
Variant: 8.1
Lookup Type: normal

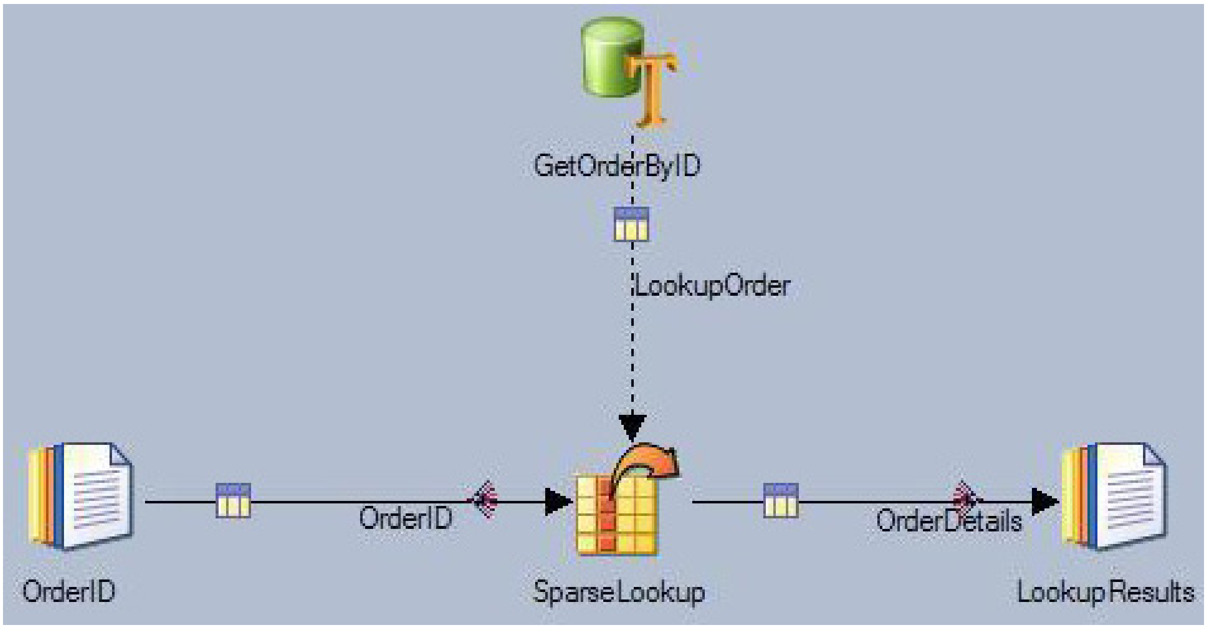
Generate SQL	Yes
Table name *	Orders
Enable quoted identifiers	No
Describe Strings In Bytes	No
Access method	Immediate
▶ SQL	
▶ Transaction	
▶ Session	
▶ Before/After SQL	No
▶ Bulk access	
▶ Limit settings	
▶ Parallel synchronization	No

Sparse lookup

For sparse lookup, a database query is generated based on each input record and the query is sent to the target database to get the result.

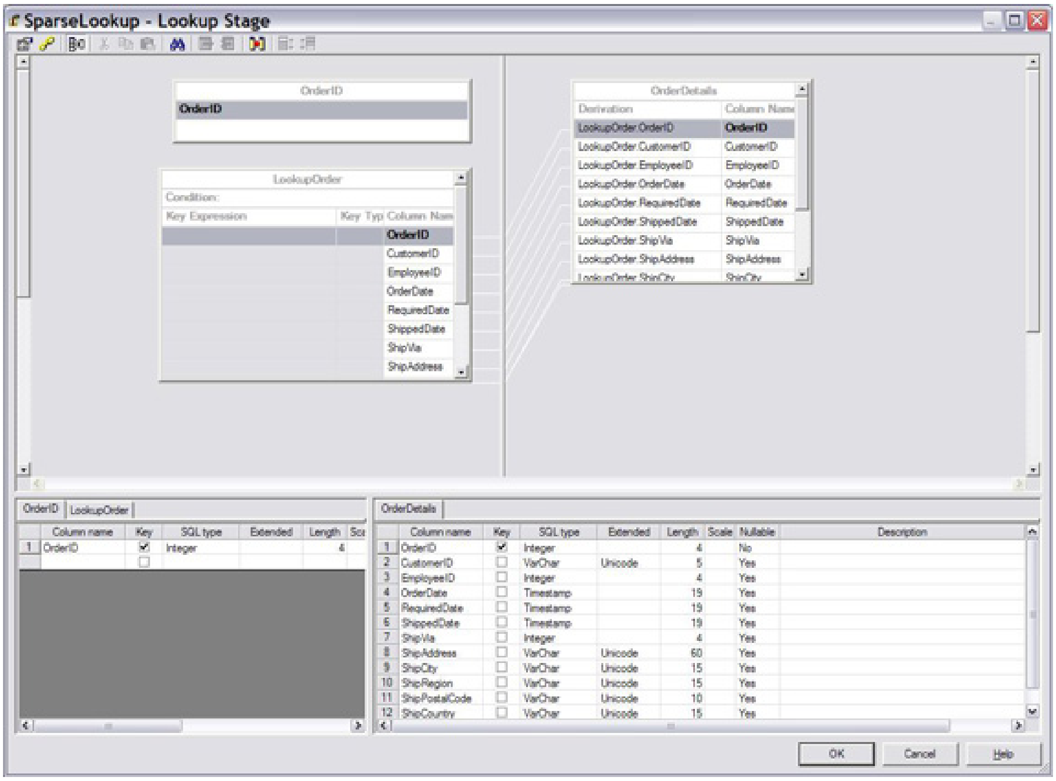
Figure 22 shows the lookup stage and the Teradata connector of a sample job to perform a sparse lookup. For each order ID, the lookup stage named SparseLookup sends the order ID to the Teradata connector named GetOrderByID. The connector queries the order detail based on the order ID and returns the query result to the lookup stage. The lookup stage forwards the query result to the output link named OrderDetails. The job performs a database query for each order ID. Since there are four order IDs, the job performs four database queries.

Figure 22. DataStage job to perform sparse lookup



Two main steps are required to perform a normal lookup:

1. As shown in Figure 23, set up the lookup stage to perform the sparse lookup.
   Figure 23. Set up the lookup stage for sparse lookup



SparseLookup - Lookup Stage

OrderID
OrderID

LookupOrder

Condition:

Key Expression	Key Typ	Column Nam
		OrderID
		CustomerID
		EmployeeID
		OrderDate
		RequiredDate
		ShippedDate
		ShipVia
		ShipAddress

OrderDetails

Derivation	Column Name
LookupOrder.OrderID	OrderID
LookupOrder.CustomerID	CustomerID
LookupOrder.EmployeeID	EmployeeID
LookupOrder.OrderDate	OrderDate
LookupOrder.RequiredDate	RequiredDate
LookupOrder.ShippedDate	ShippedDate
LookupOrder.ShipVia	ShipVia
LookupOrder.ShipAddress	ShipAddress
LookupOrder.ShipCity	ShipCity

OrderID | LookupOrder

	Column name	Key	SQL type	Extended	Length	Sca
1	OrderID	☑	Integer		4	
		□				

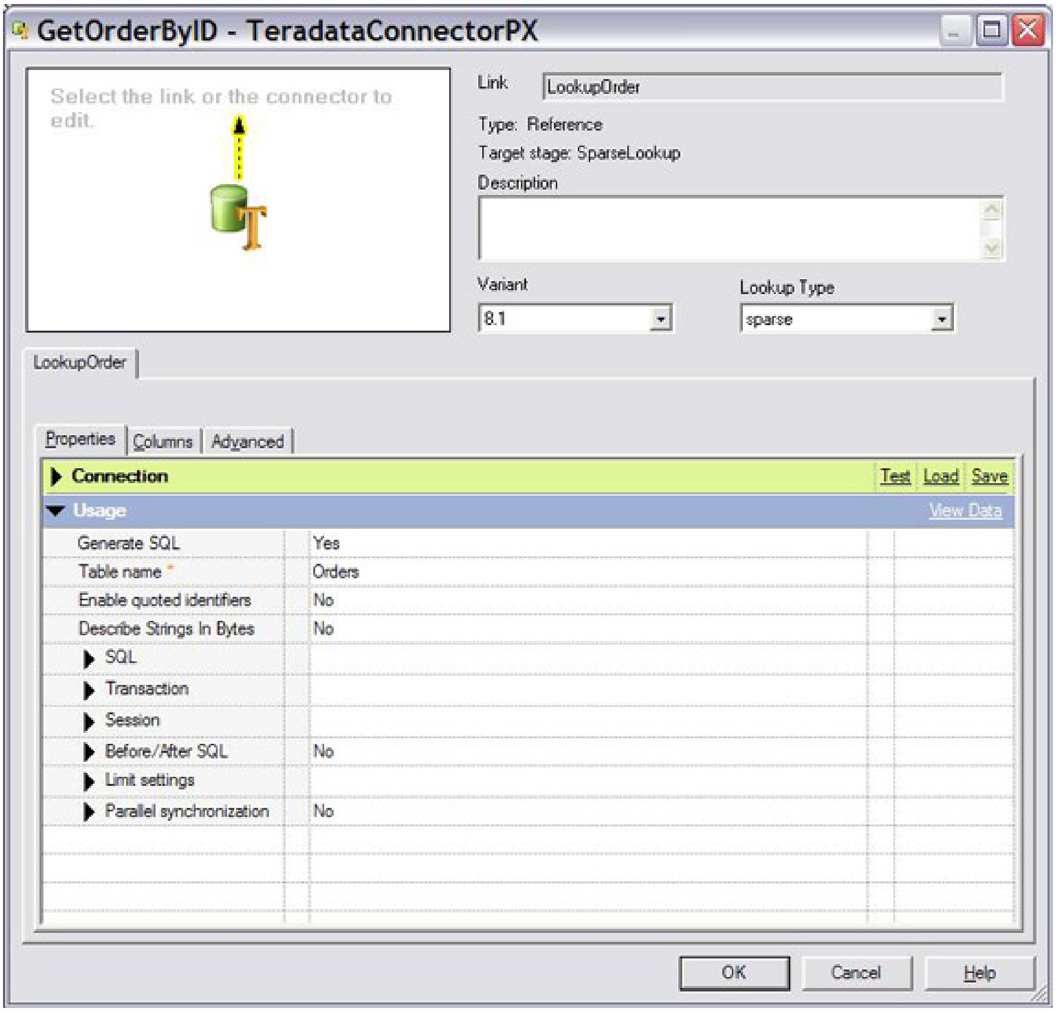
OrderDetails

	Column name	Key	SQL type	Extended	Length	Scale	Nullable	Description
1	OrderID	☑	Integer		4		No	
2	CustomerID	□	VarChar	Unicode	5		Yes	
3	EmployeeID	□	Integer		4		Yes	
4	OrderDate	□	Timestamp		19		Yes	
5	RequiredDate	□	Timestamp		19		Yes	
6	ShippedDate	□	Timestamp		19		Yes	
7	ShipVia	□	Integer		4		Yes	
8	ShipAddress	□	VarChar	Unicode	60		Yes	
9	ShipCity	□	VarChar	Unicode	15		Yes	
10	ShipRegion	□	VarChar	Unicode	15		Yes	
11	ShipPostalCode	□	VarChar	Unicode	10		Yes	
12	ShipCountry	□	VarChar	Unicode	15		Yes	

2. As shown in Figure 24, specify the following parameters to set up the Teradata connector for sparse lookup:

    - The sparse lookup type.
    - The immediate access method. The bulk access method can not be used for a sparse lookup.
    - The Orders target table and auto-generate SQL. The connector generates the query SQL at runtime based on the target table and the column definitions.

Figure 24. Set up the Teradata connector for sparse lookup



GetOrderByID - TeradataConnectorPX
Select the link or the connector to edit.
Properties
Columns
Advanced
Test
Load
Save
View Data
OK
Cancel
Help

Link: LookupOrder

Type: Reference

Target stage: SparseLookup

Description

Variant: 8.1

Lookup Type: sparse

LookupOrder

Connection

Usage

Generate SQL	Yes
Table name *	Orders
Enable quoted identifiers	No
Describe Strings In Bytes	No
SQL	
Transaction	
Session	
Before/After SQL	No
Limit settings	
Parallel synchronization	No

Legacy Teradata Enterprise stage

The Teradata Enterprise (TDEE) stage is a legacy Teradata stage available since DataStage Version 7.x. TDEE stage is a native PX-operator that provides the following features:

- It is a high performance solution for loading and exporting large amounts of data. The Teradata connector provides the equivalent functions via the load and export drivers in the bulk mode option.
- It uses the Teradata call lever interface CLIv2 and FastLoad/FastExport protocols.
- It supports the Teradata client versions 8.x, 12.x, and 13.x.
- It does not support the update, upsert, or sparse-lookup operations.

- It does not support the End of Wave and reject link features.

Figure 25 shows a sample ETL job that illustrates the TDEE data extract and load features. TDEE_Extract exports data from a Teradata database. TDEE supports data export from a table or using a user-defined SQL. TDEE_Load loads data into a Teradata table. TDEE supports loading data into a table only. User-defined SQL is not supported for the data load operation.

Figure 25. Data extract and load using TDEE

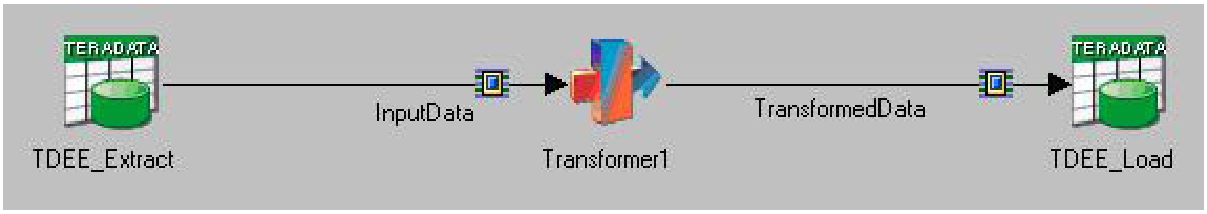
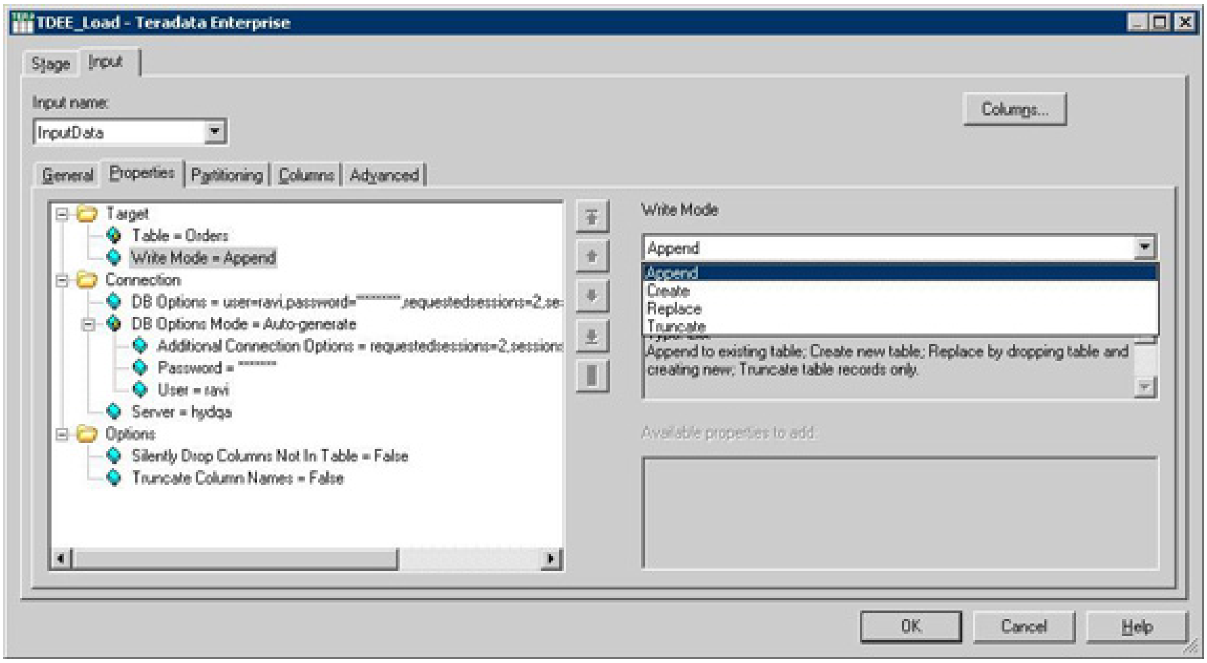


Figure 26 illustrates how to set up the TDEE_LOAD stage. TDEE can perform the following pre-load operations based on the write mode selection:

- Create — to create a new table
- Replace — to drop the existing table and then create a new table
- Truncate — to delete the records in the existing table

Figure 26. Set up the TDEE data load operation



Input name: InputData

- Target
  - Table = Orders
  - Write Mode = Append
- Connection
  - DB Options = user=ravi,password=********,requestedsessions=2,se
  - DB Options Mode = Auto-generate
    - Additional Connection Options = requestedsessions=2,sessions
    - Password = ********
    - User = ravi
  - Server = hydqa
- Options
  - Silently Drop Columns Not In Table = False
  - Truncate Column Names = False

Write Mode: Append

Append
Create
Replace
Truncate

Append to existing table; Create new table; Replace by dropping table and creating new; Truncate table records only.

Available properties to add:

The Teradata Enterprise stage uses the FastLoad protocol to load data into a table. The FastLoad protocol supports loading only into empty-tables. When the write mode Append is selected, the stage inserts data into a temporary work-table using the FastLoad protocol. After completing the data load operation, the stage inserts data into the target table using the following SQL:

```
insert into <target table> select * from <temporary work table>
```

The Teradata Enterprise stage operates in parallel mode. It supports the creation of multiple processes running on multiple computing nodes for the data load or export operation. If requestedsessions/sessionsperplayer properties are defined, they control the number of player processes spawned for the data operation. Otherwise, the default value for the number of player processes spawned for the data operation is set to half of the number of the Teradata Access Module processors.

The multiple loading processes need to be synchronized at various points during the data operation. A terasync database table is created or used for synchronization. A row is inserted into the table for each job that is run. Each player process updates the row to indicate its current status. The job aborts if all the player processes cannot be synchronized within the timeout period, which by default is 20 seconds. You can change the default by specifying `synctimeout=<`*`specified_value`*`>` as an Additional Connection Option option on TDEE data load definition (see Figure 26).

Legacy Teradata Multiload stage

The Teradata MultiLoad (TDMLoad) stage was originally designed for the DataStage server. The TDMLoad stage supports both data load and export. It internally uses the Teradata FastExport utility for export. It uses the Teradata MultiLoad or TPump utility for load. The TDMLoad stage also works on the DataStage PX. However, unlike the Teradata connector, it only runs in sequential mode. Running the TDMLoad in parallel mode is not supported.

Prior to the availability of the Teradata connector, the TDMLoad stage was mainly recommended and used for supporting the database update and upsert operations. The Teradata connector provides equivalent features via the update and stream drivers in the bulk mode option.

Figure 27 shows a sample ETL job that illustrates the TDMLOAD data export and load features. TDMLOAD_Export exports data from a Teradata database. TDMLOAD_Load_Update loads data into a Teradata table.

Figure 27. Data extract and load using TDMLOAD

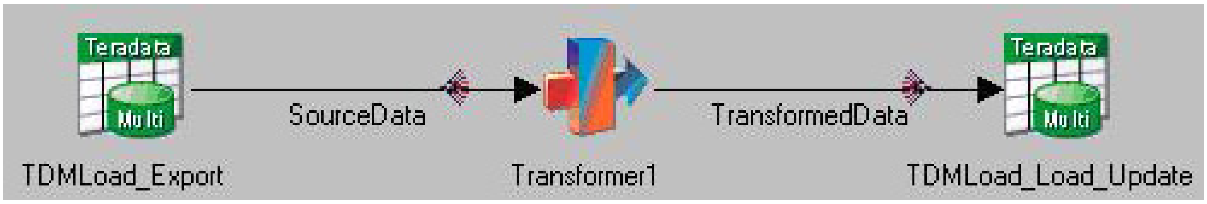
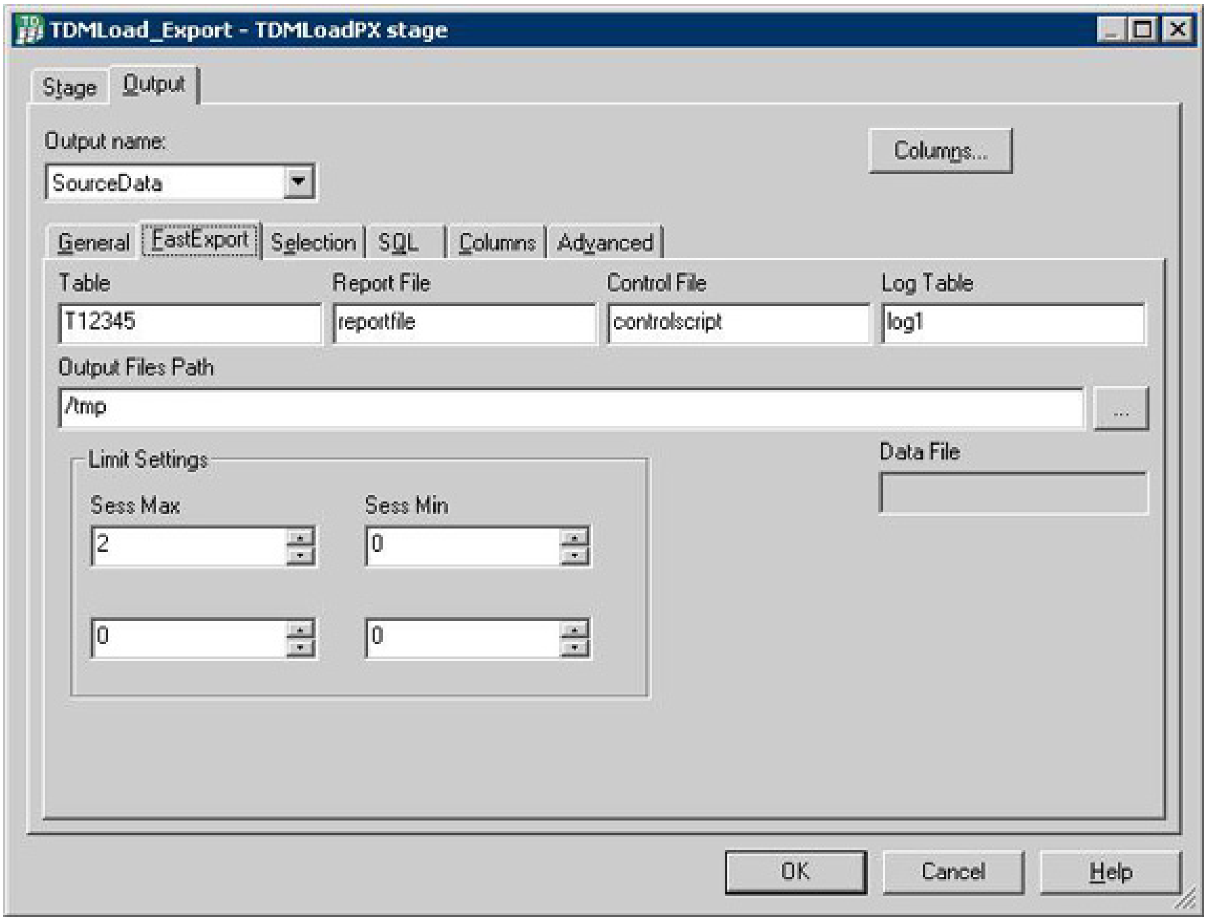


Figure 28 shows how to set up the TDMLOAD data export operation. The following describes how the TDMLoad stage works for the data export operation:

- The stage invokes the Teradata FastExport utility with the given SQL statement.
- The FastExport utility reads the data from Teradata in Teradata format and writes the data to a pipe or a data file.
- The stage reads the data from a pipe or a data file and writes the data to the output link.

Figure 28. Set up the TDMLOAD data export



Output name: SourceData

Table: T12345

Report File: reportfile

Control File: controlscript

Log Table: log1

Output Files Path: /tmp

Limit Settings

Sess Max: 2

Sess Min: 0

0

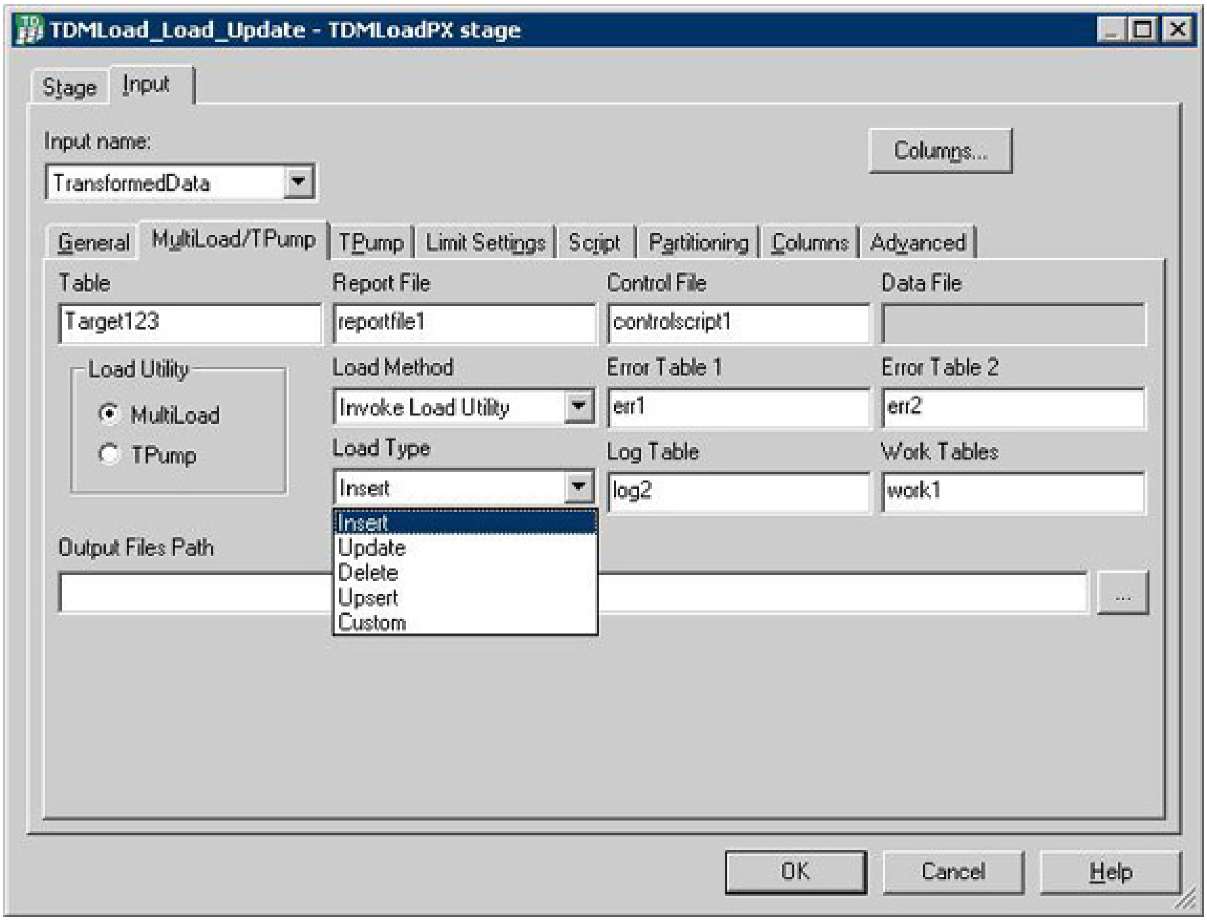
0

Data File:

Figure 29 shows how to set up the TDMLOAD data load operation. The following describes how The TDMLoad stage works for the data load operation:

- The stage reads the data from the DataStage input link.
- The stage converts the data to a Teradata format and writes to a data file or a pipe.
- The stage generates a Teradata load utility script and then invokes the Teradata MultiLoad or TPump utility based on the user selection.
- The selected Teradata utility uses the generated script as input and writes the output to a report file.

Figure 29. Set up the TDMLOAD data load



Input name: TransformedData

Table: Target123

Report File: reportfile1

Control File: controlscript1

Data File:

Load Utility: ● MultiLoad ○ TPump

Load Method: Invoke Load Utility

Error Table 1: err1

Error Table 2: err2

Load Type: Insert

Log Table: log2

Work Tables: work1

Output Files Path:

The TDMLoad stage provides the option of writing the DataStage data to a data file in the Teradata FastLoad or VarText format. You can use the Teradata load utilities to load the data file outside of the DataStage at a later time. The Teradata connector does not support this feature.

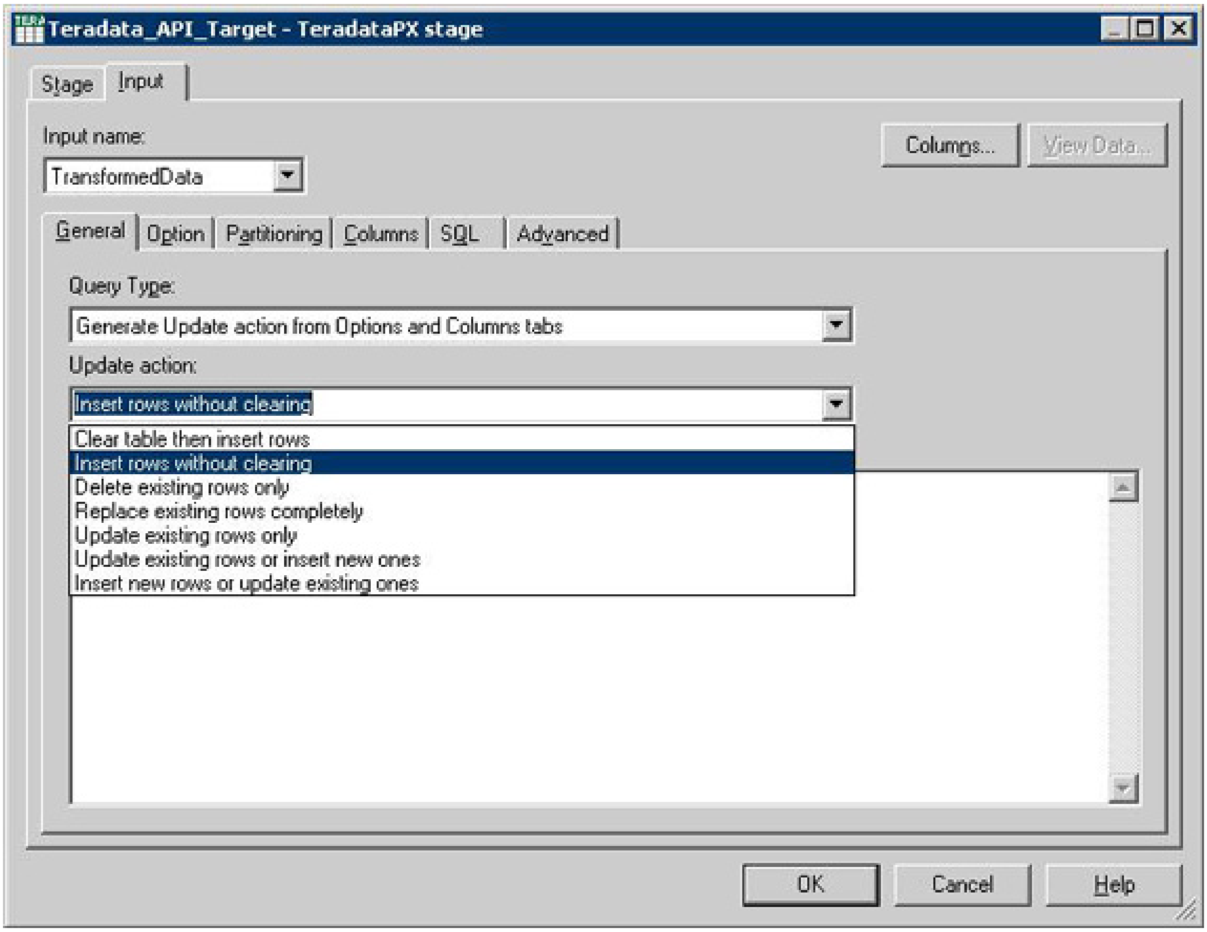
Other Legacy Teradata stages

The Teradata API (TDAPI) stage was designed for the DataStage server. It provides the functions for executing the SQL select/insert/update/upsert/delete statements via the Teradata DBC/SQL partition. It works on the DataStage PX in sequential mode. Running the stage in parallel mode is not supported.

The Teradata API stage processes one data record at a time. It does not leverage the Teradata DML array operation feature. The array operation sends many rows at once to the server. The stage was recommended for processing a small number of records. The immediate mode of the Teradata connector supports the SQL execution by the DBC/SQL partition. The connector also allows users to specify the array size to use the Teradata array operation feature.

Figure 30 shows the stage definition for the Teradata API stage being used to insert or update a database table.

Figure 30. Teradata API stage definition



Input name: TransformedData

Query Type: Generate Update action from Options and Columns tabs

Update action: Insert rows without clearing

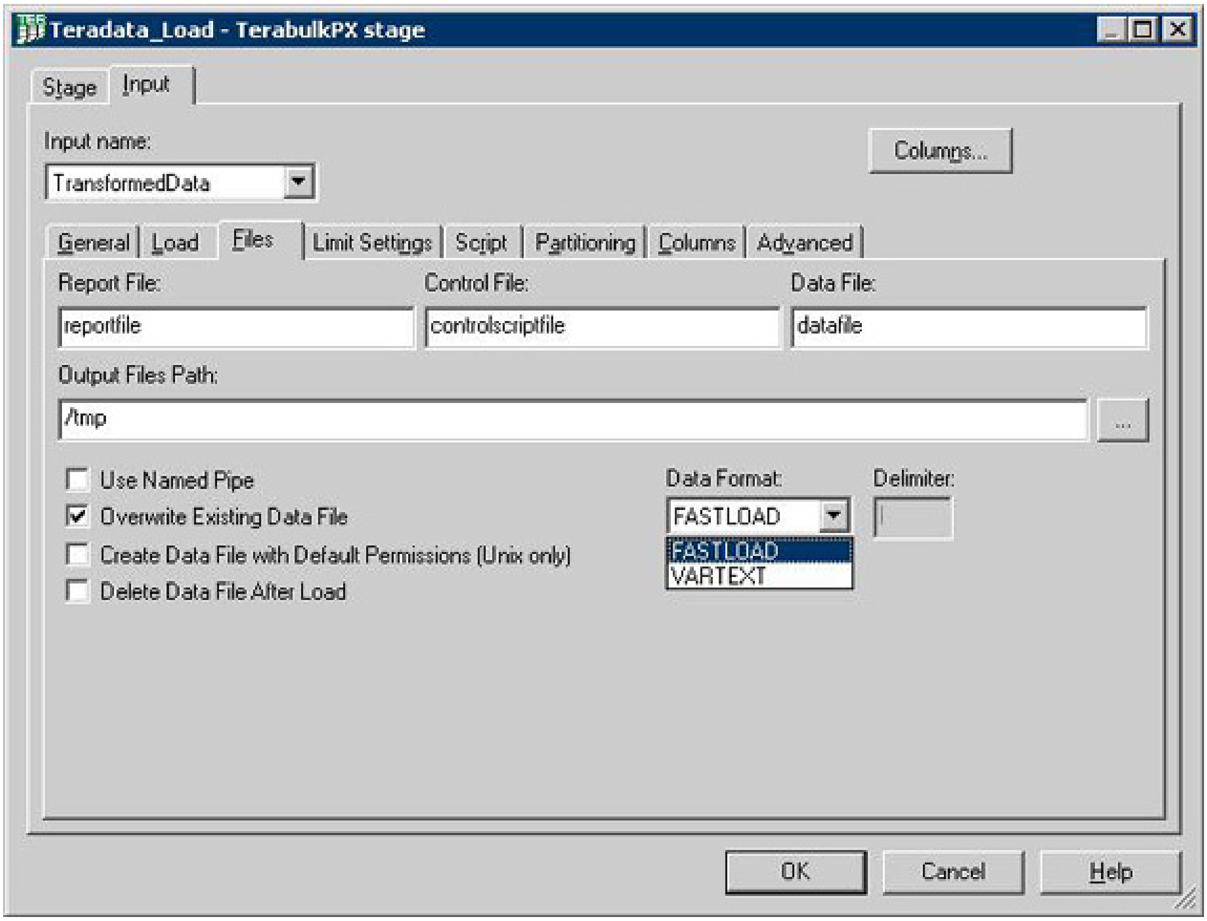
Teradata_API_Target - TeradataPX stage
Stage
Input
Columns...
View Data...
General
Option
Partitioning
Columns
SQL
Advanced
Clear table then insert rows
Insert rows without clearing
Delete existing rows only
Replace existing rows completely
Update existing rows only
Update existing rows or insert new ones
Insert new rows or update existing ones
OK
Cancel
Help

The Teradata Load (terabulk) stage was also designed for the DataStage server. It uses the FastLoad utility and provides the function of loading bulk data into an empty database table. It works on the DataStage PX in sequential mode. Running the Teradata load stage in parallel mode is not supported. The Teradata connector provides the equivalent load feature via the load driver in the bulk mode option.

The Teradata Load stage provides the option of writing the DataStage data to a data file in the Teradata FastLoad or VarText format. The Teradata connector does not support this feature.

Figure 31 shows the stage definition for the Teradata Load stage when the stage is used to load data into a database table.

Figure 31. Teradata Load stage definition



Input name: TransformedData

Report File: reportfile

Control File: controlscriptfile

Data File: datafile

Output Files Path: /tmp

☐ Use Named Pipe

☑ Overwrite Existing Data File

☐ Create Data File with Default Permissions (Unix only)

☐ Delete Data File After Load

Data Format: FASTLOAD

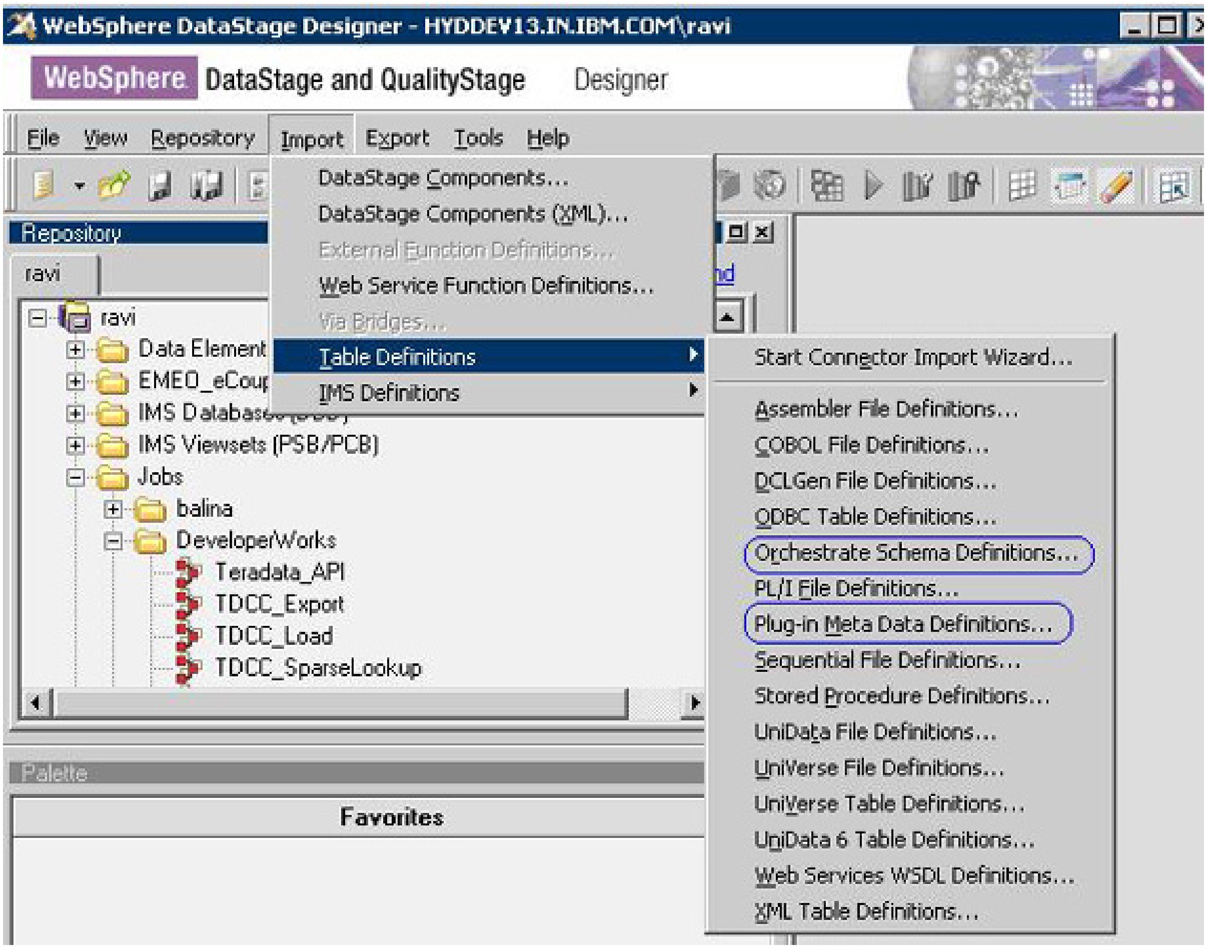
Delimiter:

Legacy metadata import services

As shown in Figure 32, you can invoke the legacy metadata import services via these menu items:

- Orchestrate Schema Definitions
- Plug-in Meta Data Definitions

Figure 32. Legacy metadata import options


WebSphere DataStage Designer - HYDDEV13.IN.IBM.COM\ravi
WebSphere DataStage and QualityStage Designer
File View Repository Import Export Tools Help
Repository
ravi
ravi
Data Element
EMEO_eCoup
IMS Databases
IMS Viewsets (PSB/PCB)
Jobs
balina
DeveloperWorks
Teradata_API
TDCC_Export
TDCC_Load
TDCC_SparseLookup
Palette
Favorites
DataStage Components...
DataStage Components (XML)...
External Function Definitions...
Web Service Function Definitions...
Via Bridges...
Table Definitions
IMS Definitions
Start Connector Import Wizard...
Assembler File Definitions...
COBOL File Definitions...
DCLGen File Definitions...
ODBC Table Definitions...
Orchestrate Schema Definitions...
PL/I File Definitions...
Plug-in Meta Data Definitions...
Sequential File Definitions...
Stored Procedure Definitions...
UniData File Definitions...
UniVerse File Definitions...
UniVerse Table Definitions...
UniData 6 Table Definitions...
Web Services WSDL Definitions...
XML Table Definitions...

Selecting the Orchestrate Schema Definitions menu item starts the process of importing the file definition or database table schema using the PX operators. When the Teradata database type is selected, the process invokes the Teradata enterprise stage (TDEE) to import the selected Teradata table schema.

Selecting the Plug-in Meta Data Definitions menu item also starts the process of importing database table schema using the DataStage plug-in stages. When the Teradata database type is selected, the process invokes the Teradata API stage to import the selected Teradata table schema.

Conclusion

This article demonstrates how to integrate Teradata data with other data sources using the Teradata connectivity solutions within the IBM InfoSphere Information Server. It introduces the data loading, data extraction, and lookup features of the Teradata connector. It also explains the main features of the Teradata legacy stages. The Teradata connector provides a single solution to replace all the legacy stages. Many examples are given to illustrate the step-by-step design processes.

The IBM InfoSphere Information Server provides leading technology and integration solutions to address many critical data integration issues, including:

- Data Quality. The data that builds a data warehouse often comes from various data sources. The structure of the legacy data is often not documented and the data quality is poor. The InfoSphere Information Analyzer analyzes your data and determines the data structure and quality. It helps you understand your data. The InfoSphere QualityStage solutions standardize and match any type of information to create high quality data.
- Data Volume. There is often a huge amount of data that needs to be processed regularly in a data warehouse environment. Sometime the data volume grows beyond expectations. The issue needs to be addressed with a scalable ETL architecture. The IBM InfoSphere Information Server leverages the pipeline and partition technologies to support high data throughput. It can be deployed on SMP (Symmetric Multiprocessing) and MPP (Massively Parallel Processing) computer systems to achieve maximum scalability.

Acknowledgements

The authors would like to thank Tom O’Shea and Stewart Hanna for their reviews of this paper and Ravi Nandibhatla, Srinivas Mudigonda, and Vikram Vangari for their discussions on the STP and ODBC stages.

Resources

- Grow your InfoSphere skills at the InfoSphere page on developerWorks
- Get more details from the IBM Information Server information center.
- Learn more about Teradata database software.

About the authors

Jeff Li
Jeff Li is a software architect of Software Group in Boca Raton, Fla. He has been working in the area of ETL, enterprise data integration, and other applications for 15 years. He is also a certified project management professional.

Ravi Tatapudi
Ravi Tatapudi is an Advisory Software Engineer working in India Software Lab, Hyd, India. He joined IBM India in early 2006 and has worked on the Teradata components of Information Server. Prior to joining IBM, he worked with Teradata as a contractor for 7 years. Prior to that, Ravi worked in the PC hardware unit in ECIL (a Government of India enterprise) for 8 years.